Computer Science • 21(2) 2020 https://doi.org/10.7494/csci.2020.21.2.3356

Borislava Vrigazova
Ivan Ivanov

# TENFOLD BOOTSTRAP PROCEDURE FOR SUPPORT VECTOR MACHINES

**Abstract** *Cross validation is often used to split input data into training and test sets in support vector machines. The two most commonly used cross validation versions are tenfold and leave-one-out cross validation. Another commonly used resampling method is random test/train split. The advantage of these methods is that they avoid overfitting in a model and perform model selection. However, they can increase the computational time for fitting support vector machines by increasing the size of the dataset. In this research, we propose an alternative for fitting SVM, which we call tenfold bootstrap for support vector machines. This resampling procedure can significantly reduce execution time despite the large number of observations while preserving a model's accuracy. With this finding, we propose a solution to the problem of slow execution time when fitting support vector machines on big datasets.*

**Keywords** support vector machines, bootstrap, cross validation

**Citation** Computer Science 21(2) 2020: 253–268

## 1. Introduction

Tenfold cross validation is used in support vector machines (SVMs) [10] to select the best C tuning parameter to balance accuracy and smoothness [27]. Statistical software programs like R and Python use a function grid search where researchers input an initial range of values for C (among which cross validation chooses the best values for C). The program fits support vector machines with each value of C and then chooses the value resulting in the best balance between smoothness and accuracy. If the value of the penalty parameter C is too big or too small, bias can be inserted into a model (which can affect the accuracy scores). Moreover, cross validation often makes the support vector classifier computationally exhaustive despite avoiding overfitting. In this research, we propose an alternative method for fitting SVM that can improve the results from support vector machines in numerous ways. First, our method can significantly reduce the time needed for fitting SVM and prediction. If cross validation fits SVM for several hours in big datasets, our method can reduce the computational time to less than an hour. Second, our proposition (like cross validation) avoids overfitting, as it resamples observations in the test and training sets at each iteration. Third, the resampling procedure allows us to keep the high accuracy of SVM. In fact, our method improved the accuracy scores of SVM in some cases as compared to cross validation. Our experiments in Python 3.6 also showed that our algorithm can reduce the bias coming from the value of the C tuning parameter as we fix it. Our algorithm does not require preliminary transformations of the target and independent variables. At the same time, it significantly reduces computational time, bias, and overfitting while improving the accuracy of the support vector classifier. We call our procedure tenfold bootstrap.

## 2. Literature review

Support vector machines are supervised learning models that classify new observations into one of the existing classes in a dataset. Cross validation is used in order to increase the prediction ability of the model to classify new observations. Tenfold cross validation uses all observations from a training set in order to grasp the characteristics of the dataset and then predicts to which class the observations from the test set will belong [27]. The resulting advantages of cross validation include the avoidance of selection bias and overfitting. These advantages have made it widely used in machine-learning methods for feature-selection, classification, and regression.

In feature-selection methods, cross validation finds the smallest value of a tuning parameter to minimize the prediction error of a model so that the optimal number of parameters can be shrunk down to zero [11], [28], and [21]. In classification problems, tenfold cross validation may be used to select the most appropriate method for prediction (as in [2] and [3]). In other cases, it can be a tool for predicting time series [6]. In regression problems, cross validation can be used as a model-selection technique instead of Mallow's criteria [19].

Despite the practical advantages of cross validation, some authors have shown [9] that tenfold cross validation in classification problems may be outperformed by its smoothed version. In datasets with fixed regressors, feature-selection methods [11, 28], and [21] with cross validation all fail to shrink the coefficients before all insignificant variables to zero. Breiman [5] showed that the reason for this failure was cross validation. He showed that the little bootstrap procedure [22] could capture the characteristics of fixed regressors better than cross validation and proposed that fixed regressors should be used with the little bootstrap procedure while the random effects can be captured by cross validation. Later, Vrigazova [22] showed that Breiman's contribution can be extended to panel data. She showed that, in panel datasets with fixed regressors, feature-selection methods with bootstrap can be as competitive as econometric panel methods are. In panel data, bootstrap can also be used for detecting causality between cross-sectional units [26] and time series [20]. In economic time series data, bootstrap has recently been used [29] to detect non-stationary seasonality.

The advantages of the bootstrap procedure were also recognized by MacKinnon [16] in Monte Carlo simulations. He proposed that, unlike cross validation, bootstrap can be used to build confidence intervals based on resampling with repetition. The resulting distribution of the sample can approximate to the Gaussian distribution, which makes bootstrap widely used in Monte Carlo simulations. The underlying unknown distribution of data can also be inferred. Despite the numerous advantages of bootstrap, it has not been recognized as a standard resampling procedure in support vector machines. Unlike cross validation, bootstrap allows for random sampling with repetition. In this paper, we show that this attribute of bootstrap has important practical implications for fitting support vector machines.

In academic literature and machine-learning textbooks, support vector machines are fitted by cross validation [12] to avoid overfitting and selection bias. Support vectors define a hyperplane that divides data into classes, and cross validation increases the prediction ability of a model to classify each class. Many data scientists experience a practical problem when they run SVMs with cross validation on a dataset with a large number of features. The issue [23] with cross validation in this case is the very slow computational time, which increases proportionally to the number of features in a dataset. As the features increase, the computing time can take from a couple of hours to a couple of days. Some researchers ( [17], [13]) proposed algorithms for SVMs that reduce the number of features to accelerate the computing time. Although efficient, these algorithms can be difficult to interpret. Moreover, each dimensionality reduction algorithm suggests a different set of variables to be used in SVMs, which can result in controversy. Recent studies ( [15], [18], and [24]) have shown that the boostrap procedure may have a number of benefits in classification problems, like reducing a model's uncertainty or finding the appropriate number of clusters in the k-nearest neighbor.

The aim of our research is to propose a solution using the bootstrap procedure that can significantly reduce computing time in the case of a large dataset without

reducing the size of the dataset. With this research, we extend the practical applications of bootstrap in the classic C-support vector machines problem. Section 3 presents a theoretical description of our findings. Section 4 describes our practical results, and Section 5 concludes the paper.

## 3. Methodology

The support vector classifier divides a dataset into classes based on a hyperplane that is defined by a training set. It then classifies each test observation into the class to which it belongs. The support vector machine aims at maximizing the value of the M margin in order to correctly classify each observation. The greater the margin, the more precise is the classification. In this paper, we try to solve the C-optimization problem for the support vector machines formulated in [7]. The C-SVM contains a nonnegative C tuning parameter, which can be selected via cross validation [12]. As there is a trade-off between smoothness and accuracy, the values of the C tuning parameter should be optimal in terms of balancing between them. Researchers provide a grid of input values for C, and cross validation selects the C estimator that results in the best balance between smooth decision boundaries and the correct classification of the training observations. Depending on the datasets and the goal of the research, the initial values in the grid can be selected in various ways. Some authors propose initial default values as in [12]; however, these may not be suitable for all types of data or the goal of the research. Using the same default values in the grid in different research pieces may increase the bias in the model. In practice, the choice of the best C tuning parameter can be made in software products like R and Python via the grid search function.

In this paper, we fix $C = 1$ for all of the algorithms we run. We fix $C = 1$ for the widely used algorithms (1–3) as well as for the one we propose. Therefore, we can avoid the bias that comes from changing the grid values. As we aim to improve the computing time of fitting C-support vector machines while maintaining accuracy, we compare our results to widely used resampling algorithms for SVC. The standard methods that we use for comparison are tenfold cross validation (Algorithm 1), leave-one-out cross validation (Algorithm 2), and train/test random splitting (Algorithm 3). These methods are well-known resampling methods that are used to avoid overfitting and increase the prediction accuracy of the support vector classifier. We describe the steps we follow to run the standard algorithms in details in the algorithms (1–3). Algorithm 4 describes the necessary steps for running the tenfold bootstrap procedure we propose to optimize the performance of the support vector classifier. The bootstrap procedure we propose is not a novel resampling method; however, we propose a novel application of bootstrap that can significantly boost the performance of the support vector classifier and produce a comparable accuracy to the standard resampling methods. We compare the performance of the four algorithms by calculating the accuracy, precision, recall, $f_1$-score, and computational time. These metrics are also standard for evaluating the performance of the support vector classifier [12].

We use the following abbreviations throughout this and the following sections:

- support vector classifier – SVC,
- cross validation – cv,
- bootstrap – btsp,
- tuning parameter – C,
- leave-one-out cross validation – looCV.

*Classic Algorithm 1: Tenfold cross validation for data resampling*

1. We load in Python 3.6 datasets of different sizes. We describe our data in the next section and present a link to them. We define the target variable and independent variables in each dataset. The target variable in all datasets is categorical. We do not standardize the variables, as we examine the performance of all of the algorithms.
2. We apply a rule for splitting each dataset into training and test sets (at a 70/30 proportion) by using the train_test_split function from the model_selection module in Python 3.6. We keep all of the other settings in the train_test_split function by default.
3. After we have defined the splitting rule, we apply tenfold cross validation to randomly choose observations that participate in the training and test sets. Cross validation [12] and [14] is commonly used in practice to avoid overfitting and to balance between unbiasedness and variance. Tenfold cross validation is a special case of K-fold cross validation, where K is the number of samples drawn from the dataset, in our case, K = 10).
4. We used tenfold cross validation to produce ten samples (folds) from each dataset; each sample contains different observations in the training and test sets. We use the sample from the I-th fold as the test observations. All of the training and test sets contain observations without replacement.
5. We run tenfold cross validation by using the cross_validation.kfold module with K=10, and we keep the other options by default. An important difference between our algorithm and that found in [12] is that we do not use cross validation to find C in the support vector classifier. We use cross validation to resample the observations in the test and training sets. Rather than finding the best value of C to improve the accuracy [12], we look for the representative training and test samples that result in high accuracy. We do this via tenfold cross validation.
6. We run the C-support vector classifier [7] on the training sample from each cv fold and evaluate its performance on the I-th fold. To fit the support vector classifier, we use the sklearn.SVC.svm module. The module has a default value for C = 1, and the shrinkage option in the SVC.svm module is true by default. We fixed C = 1 and kept shrinkage = TRUE, as we compared the influence of resampling the training and test sets on the performance of SVC and choose the best one based on the time and accuracy. In practice, we fit the support vector classifier ten times on different training data and then validate the classifier on a different test set each time.

7. In order to validate the prediction ability of SVC at each fold, we record the time required for computations as well as the accuracy of the support vector classifier. We then average the time and accuracy from tenfold cv and compare the results with our alternative Algorithms 2, 3, and 4.

*Classic Algorithm 2: Leave-one-out cross validation for resampling data*

1. In the first step, we load the same datasets as in Algorithm 1 in Python 3.6. We define the target and independent variables for each dataset without making any additional transformations to the data.
2. We apply a rule for splitting each dataset into training and test sets (at a proportion of 70/30) by using the train_test_split function from the model_selection module in Python 3.6. We keep all of the other settings in the train_test_split function by default.
3. We then select the observations from each dataset that will be part of the training and test sets based on leave-one-out cross validation [25].
4. Unlike tenfold cv (where K = 10), looCV fixes K to be equal to the number of observations (N) in each dataset. In our experiments, K = N, and for each dataset, we run K = N iterations. In Python 3.6, we use the sklearn.model_selection.LeavePOut(p) class to run looCV. This class allows the researcher to run not only looCV but also leave-two-out cv and etc by modifying the p parameter. The p parameter defines the number of observations from each dataset that we use for the test set. The rest of the N-p observations are used as the training set. In sklearn.model_selection.LeavePOut(p), we fix p = 1 so that the test set for each dataset consists of only one observation while the training set contains N-p observations.
5. After we have defined p, we run the script N times for each dataset; at each iteration, we get different training and test sets. As the looCV does not allow resampling with replacement, the training and test set do not contain repeated observations. Each training set consists of the rest of the N-p observations.
6. We then run the support vector classifier on each training set and validate its performance by the test set. We record the time for performance as well as the accuracy of each iteration and average the results. We then compare them to the rest of the algorithms. The advantage of leave-one-out cross validation is its higher accuracy, while tenfold cross validation can reduce overfitting [12] and [14].

*Classic Algorithm 3: repeated random test-train splits*

1. We first load our data into Python 3.6 and define the target and independent variables. We do not perform any transformations to the data.
2. We define the splitting rule for the test and training sets as 70/30 (as in [12]).
3. We then apply the repeated random test-train splits [14] to form the training and test sets based on the 70/30 rule. This technique involves splitting the dataset into training and test sets using the splitting proportion by randomly choosing ten different subsamples for the training and test sets. Unlike cross validation, the repeated random test-train split allows for resampling with and without re-

placement. We used the ShuffleSplit function part of the model_selection module in Python 3.6 and ran ten splits. We tested this method, as some authors consider it to be faster than leave-one-out and tenfold cross validation [14].

4. We run the SVC method on each training and test sample formed in the previous step.
5. We then average the time and accuracy of the models and compare them to Algorithms 1 and 2.

*Proposed Algorithm 4: Tenfold bootstrap*

As our experiments with the three types of cross validation did prove to be computationally challenging, we decided to test a fourth resampling method. This is the bootstrap procedure that can be found in [8]. The procedure consists of several steps:

1. We loaded the data into Python 3.6 and defined the target variable as well as the independent variables. We did not perform any additional transformations of the data.
2. We defined the rule for splitting each dataset into training and test sets as 30/70 (unlike in Algorithms 1–3).
3. We followed the procedure in [8] and ran the bootstrap procedure in Python 3.6; however, instead of using bootstrapped module [1], we created our script. In our script, replacement is allowed when forming the test and training sets. Unlike other resampling methods where the training and test sets contain unique observations from the dataset, our algorithm allows for the repetition of observations in the training set. When the bootstrap procedure splits the data into a training set using 30 percent of the observations, the training set can contain one observation more than once. The test set, however, contains observations without repetition. This is the main difference between the bootstrap procedure and Algorithms 1–3.
4. We then run the support vector classifier on each training and test set.
5. We then averaged the time and accuracy and compared the results to Algorithms 1, 2, and 3.

Although some authors [4] suggest that the bootstrap procedure should be run 1,000 times, we decided to run 10 folds for each dataset. As the number of folds increases in the bootstrap procedure, the required computational time increases. Our experiments show that fitting SVM with 'a tenfold bootstrap' is enough to provide accuracy results comparable to the three types of cross validation while significantly reducing computational time. We called bootstrap with ten samples a tenfold bootstrap. As Section 4 shows, this finding introduces a new practical advantage of the bootstrap procedure and can be used as an alternative resampling method for improved performance in SVC.

The support vector classifier divides the dataset into classes based on a hyperplane defined by the training set. It then classifies each test observation into the class to which it belongs. The support vector machine aims at maximizing the value of the

M margin in order to correctly classify each observation. The greater the margin, the more precise is the classification. In this paper, we try to solve the C-optimization problem for support vector machines formulated in [7]. The C-SVM contains a non-negative C tuning parameter, which can be selected via cross validation [12]. As there is a trade-off between smoothness and accuracy, the values of the C tuning parameter should be optimal in terms of balancing between them. Researchers provide a grid of input values for C, and cross validation selects the C estimator that results in the best balance between smooth decision boundaries and the correct classification of the training observations.

Depending on the datasets as well as the goal of the research, the initial values in the grid can be selected in various ways. Some authors propose initial default values as in [12]; however, these may not be suitable for all types of data or the goal of the research. Using the same default values in the grid in different research pieces may increase the bias in the model. In practice, the choice of the best C tuning parameter can be made in software products like R and Python via the grid search function. In this paper, we fix $C = 1$ for all of the algorithms we run. We fix $C = 1$ for the widely used algorithms (1–3) as well as for the one we propose (4). Therefore, we can avoid the bias that comes from changing the grid values.

As we aim to improve the computing time of fitting C-support vector machines while maintaining accuracy, we compare our results to the widely used resampling algorithms for SVC. The standard methods that we use for comparison are tenfold cross validation (Algorithm 1), leave-one-out cross validation (Algorithm 2), and train/test random splitting (Algorithm 3). These methods are well-known resampling methods used to avoid overfitting and increase the prediction accuracy of the support vector classifier. We describe the steps we follow to run the standard algorithms in details in Algorithms 1–3. Algorithm 4 describes the steps to run the tenfold bootstrap procedure we propose to optimize the performance of the support vector classifier. The bootstrap procedure we propose is not a novel resampling method; however, we propose a novel application of bootstrap that can significantly boost the performance of the support vector classifier and produce comparable accuracy to the standard resampling methods.

## 4. Results

### 4.1. Datasets

We have compared the performance of the support vector classifier on nine datasets. Table 1 describes the size, predicted variable, and sources of our data. All of the data used in our research is freely available at www.kaggle.com. As Table 1 shows, we have conducted experiments with datasets of different sizes. The predicted variable in all of the datasets is categorical, and the number of observations is greater than the number of predictors ($n > p$). The independent variables contain numerical and categorical variables. Column Y shows the name of the dependent variable in the respective dataset.

**Table 1**
Datasets

| Dataset | n | p | Y |
|---|---|---|---|
| glass | 175 | 9 | Type |
| leaf | 286 | 7 | arch |
| wells | 3020 | 4 | association |
| fraud | 3255 | 4 | IsFraud |
| abalone | 4177 | 8 | Rings |
| ed | 5785 | 5 | difference |
| monica | 6367 | 11 | outcome |
| food | 23971 | 5 | sex |
| adult | 45222 | 13 | income |

Source: www.kaggle.com

## 4.2. Computing time

We first explore the computing time needed for fitting SVM using the classic approaches (Algorithms 1–3) versus our approach (Algorithm 4). Table 2 summarizes the computational times in seconds.

**Table 2**
Comparison of execution times of Algorithms 1–4

| Dataset | n | p | Algorithm 1 | Algorithm 4 | Algorithm 2 | Algorithm 3 |
|---|---|---|---|---|---|---|
| glass | 175 | 9 | 0.02 | 0.00 | 0.26 | 0.02 |
| leaf | 286 | 7 | 0.04 | 0.00 | 0.46 | 0.02 |
| wells | 3020 | 4 | 0.56 | 0.32 | 358.14 | 0.98 |
| fraud | 3255 | 4 | 3.01 | 0.59 | 726.29 | 2.12 |
| abalone | 4177 | 8 | 2.51 | 0.64 | 807.09 | 1.70 |
| ed | 5785 | 5 | 9.15 | 1.86 | 4623.18 | 6.43 |
| monica | 6367 | 11 | 5.21 | 0.93 | 2628.48 | 3.50 |
| food | 23971 | 5 | 763.03 | 30.33 | >28800 | 337.05 |
| adult | 45222 | 13 | 2434.4 | 528.61 | >28800 | 1757.79 |

Source: authors' calculations

As Table 2 shows, leave-one-out cross validation (Algorithm 2) requires the longest computing time. In small datasets, it fitted SVM ten times longer than tenfold cross validation and random test/training split. As the dataset grows, Algorithm 2 proved to be the slowest resampling method for SVM. For example, in a dataset with $n = 45222$ and $p = 13$, Algorithm 2 took more than eight hours to fit SVM compared to 40 minutes for Algorithm 1, 29 minutes for Algorithm 3, and 9.20 minutes for Algorithm 4.

Random training/test split (Algorithm 3) is a faster method than leave-one-out cross validation (Algorithm 2) and tenfold cross validation (Algorithm 1). When applied to a dataset with n = 23971 and p = 5, Algorithm 3 fitted SVM in 6 minutes compared to more than 8 hours for Algorithm 2 and 12 minutes for Algorithm 1. In this dataset, bootstrap (Algorithm 4) proved to be the fastest method, performing calculations for just 30.33 seconds! Our experiments showed that the bootstrap procedure outperformed the other three resampling methods in large and small datasets. Depending on the dimensions of the dataset, the bootstrap procedure can produce classification metrics immediately (as in the monica dataset). The bootstrap procedure fitted SVM immediately, so Python reported a computing time of 0.93 seconds, while tenfold cross validation required 5.21 seconds, and random training/test split – 3.50 seconds.

In medium-sized data like the wells dataset, bootstrap performs the calculations for 0.32 seconds, which is almost twice as fast as tenfold cross validation, 1119 times faster than leave-one-out cross validation, and 3 times faster than random test/training split. Although the random training/test split method was faster than the two versions of cross validation, it was slower than bootstrap. As Table 2 shows, classic Algorithms 2 and 3 proved to be slower than the commonly used Algorithm 1 and our proposition (Algorithm 4). A key finding from Table 2 is that the bootstrap procedure (Algorithm 4) can be a time-saving alternative to cross validation and its versions (Algorithms 1–3) for fitting SVM. As the first algorithm is the standard procedure for fitting SVM (outperforming Algorithms 2 and 3), we further compare tenfold cross validation to tenfold bootstrap.

Table 3 shows the computational time needed for fitting and prediction using Algorithms 1 and 4.

**Table 3**
Time for fitting SVM: algorithms 1 vs 4

| | | | Time for fitting SVM (s) | | Time for prediction (s) | |
|---|---|---|---|---|---|---|
| Dataset | n | p | Algorithm 1 | Algorithm 4 | Algorithm 1 | Algorithm 4 |
| glass | 175 | 9 | 0.01 | 0.00 | 0.02 | 0.00 |
| leaf | 286 | 7 | 0.04 | 0.00 | 0.04 | 0.001 |
| wells | 3020 | 4 | 0.56 | 0.32 | 3.93 | 0.52 |
| fraud | 3255 | 4 | 3.00 | 0.59 | 6.04 | 1.52 |
| abalone | 4177 | 8 | 2.51 | 0.64 | 5.06 | 1.59 |
| ed | 5785 | 5 | 9.15 | 1.86 | 18.47 | 3.76 |
| monica | 6367 | 11 | 5.21 | 0.93 | 10.41 | 1.11 |
| food | 23971 | 5 | 763 | 30.33 | 1531 | 62.8 |
| adult | 45222 | 13 | 2434 | 528 | 4785 | 301 |

Source: authors' calculations

Table 3 shows that the computing time depends on the size of the dataset. A large number of variables and/or features can increase the computing time. In large datasets, the support vector classifier with both cross validation and the bootstrap procedure requires more time for computing. Another important observation is that prediction requires more time than fitting regardless of the size of the sample.

Table 3 shows that, depending on the sample size, fitting SVM with cross validation can take between 0.016 seconds and 2,434.45 seconds. In other words, cross validation can estimate SVMs in fractions of a second with small samples; however, fitting SVM can take up to several hours as the dataset increases. However, a dataset with n = 45222 and p = 13 may not be so large. In practice, data scientists can use much larger samples, and tenfold cross validation can take several days to fit SVMs (as in the food dataset). The task of fitting SVM becomes more complicated if the kernel is not rbf but polynomial. We fitted SVM with cross validation and a polynomial of the second degree on our samples; after 8 hours, we did not gain any results. As a result, fitting SVMs with cross validation led to prolonged calculations (as Table 3 shows).

Table 3 also shows that the time for fitting SVMs with the bootstrap procedure is much less than tenfold cross validation. The computing advantage of bootstrap can be observed in small datasets (although the difference is small). However, the computing advantage of bootstrap becomes obvious as the dataset increases in size. As Table 3 shows, bootstrap fitted SVM 4.6 times faster than tenfold cross validation and predicted the classes in the test sets 16 times faster than cross validation. The fitting time for cross validation on the adult dataset took 41 minutes, while bootstrap took 9.21 minutes. Cross validation predicted the class of the observations in the test set in 80 minutes, while bootstrap took about 5 minutes. We believe that the computational advantage of the bootstrap procedure for SVMs is an important solution to accelerate prediction in large and extremely large datasets. One of the reasons for this advantage is the proportion for splitting the train/test subsets (as we use the 30/70 rule). As Tables 1–7 show, the accuracy of each dataset remains unchanged when compared to tenfold cross validation despite the small training set. However, the computing time decreased significantly.

We also check the reliability of bootstrap compared to tenfold cross validation. In the next subsection, we compare the classification metrics of the datasets resulting from Algorithms 1 and 4. This analysis is important, as the computational advantage of bootstrap can only be valid if bootstrap provides similar metrics to cross validation.

## 4.3. Comparison of classification metrics

Table 4 shows the classification metrics for the leaf dataset, which we consider to be a small dataset.

As Table 4 (the leaf dataset) shows, the classification metrics of Algorithms 1 and 4 are similar. However, Algorithm 4 increased the accuracy of the classification method like what is shown in Table 4, where bootstrap increased the accuracy measures after the second sign.

**Table 4**
Leaf dataset: Algorithm 1 vs Algorithm 4

| Algorithm 1 | precision | recall | f1-score |
|---|---|---|---|
| 0 | 0.578002 | 0.720596 | 0.641470 |
| 1 | 0.425029 | 0.281910 | 0.338983 |
| total/avg | 0.513318 | 0.535099 | 0.513564 |
| Algorithm 2 | precision | recall | f1-score |
| 0 | 0.580359 | 0.724699 | 0.644363 |
| 1 | 0.433082 | 0.286204 | 0.344077 |
| total/avg | 0.518048 | 0.538991 | 0.517215 |

Source: authors' calculations

A similar finding can be done for the other small datasets, like the glass, wells, fraud, abalone, ed, and monica datasets. An important note is that we compared tenfold cross validation to the tenfold bootstrap procedure in all of our experiments. Although the bootstrap procedure is typically used by subtracting 1000 subsamples, our experiments showed that the accuracy measures do not change significantly whether we had 1000 samples or 10 samples. This is why we decided to subtract ten different samples so that the two algorithms can be comparable. As the leaf dataset is a small dataset, we also analyze the results for a medium-sized set (like the food dataset). Table 5 shows the accuracy measures for the ed dataset, which we consider to be medium-sized.

**Table 5**
Accuracy measures for SVM: food dataset

| Algorithm 1 | precision | recall | f1-score |
|---|---|---|---|
| 0 | 0.8684528 | 1 | 0.9295957 |
| 1 | 0 | 0 | 0 |
| total/avg | 0.7542104 | 0.8684528 | 0.8073100 |
| Algorithm 4 | precision | recall | f1-score |
| 0 | 0.8671689 | 1 | 0.9288559 |
| 1 | 0 | 0 | 0 |
| total/avg | 0.7519942 | 0.8671689 | 0.8054820 |

Source: authors' calculations

In the example of the food dataset, bootstrap provided measures that were almost identical to those from tenfold cross validation (however, for a much shorter period

of time, as Tables 2 and 7 show). All classification metrics resulting from the two algorithms are similar. The case in Table 6 with the adult dataset is similar.

**Table 6**
Adult dataset: Algorithm 1 vs 4

| Algorithm 4 | precision | recall | f1-score |
|---|---|---|---|
| 0 | 0.753379 | 0.9994886 | 0.8591564 |
| 1 | 0.5641316 | 0.0019840 | 0.0039537 |
| total/avg | 0.7066438 | 0.7532098 | 0.6480114 |
| Algorithm 1 | precision | recall | f1-score |
| 0 | 0.7527646 | 0.9986182 | 0.8584353 |
| 1 | 0.5252525 | 0.0046395 | 0.0091978 |
| total/avg | 0.6963771 | 0.7522665 | 0.6479569 |

Source: authors' calculations

The adult dataset is the largest dataset in our pool of data. Table 6 shows that Algorithm 4 again resulted in increased accuracy and reduced time (Tables 2 and 7) when compared to Algorithm 1. The bootstrap procedure is able to classify each observation to the respective class with high accuracy. A key result from our research is that Algorithm 4 produces similar classification metrics to those from tenfold cross validation. An important finding from Tables 4–7 is that the bootstrap procedure fitted SVMs without a loss of accuracy as compared to the standard cross validation procedure.

**Table 7**
Algorithms 1 vs 4: summary

| Dataset | n | p | Total time for Algorithm 1 | Total time for Algorithm 4 | Average accuracy Algorithm 1 | Average accuracy Algorithm 4 |
|---|---|---|---|---|---|---|
| glass | 175 | 9 | 0.03 | 0 | 0.7 | 0.65 |
| leaf | 286 | 7 | 0.08 | 0 | 0.58 | 0.69 |
| wells | 3020 | 4 | 4.49 | 0.84 | 0.54 | 0.54 |
| fraud | 3255 | 4 | 9.05 | 2.11 | 0.65 | 0.65 |
| abalone | 4177 | 8 | 7.57 | 2.23 | 0.53 | 0.53 |
| ed | 5785 | 5 | 27.62 | 5.62 | 0.87 | 0.87 |
| monica | 6367 | 11 | 15.62 | 2.03 | 0.88 | 0.87 |
| food | 23971 | 5 | 2294.93 | 93.09 | 0.86 | 0.86 |
| adult | 45222 | 13 | 7219.93 | 830.44 | 0.75 | 0.75 |

Source: authors' calculations

Table 7 shows the total time needed (in seconds) for computing the results from the two algorithms and the average accuracy from each. The last two columns show that Algorithm 4 results in average accuracy, which is comparable to the average accuracy of Algorithm 1. Algorithm 4, however, has a significant computational advantage over Algorithm 1. The bootstrap procedure can save from between several seconds to several hours of computing time. As the table shows, the spared computing time from the bootstrap procedure is proportional to the size of the sample. The larger the sample, the greater the computational advantage of Algorithm 4. As Table 7 shows, the bootstrap procedure not only has a significant computational advantage over tenfold cross validation, but it also performs SVMs without a loss of accuracy. This finding is particularly important, as it validates the tenfold bootstrap procedure as a reliable alternative to cross validation in SVMs.

Moreover, bootstrap preserved high accuracy despite the small number of observations in the training set (30 percent). This is an important discovery, as the academic literature recommends training/test splits in 70/30, 60/40, or 50/50 proportions. Our experiments show that, as a resampling method, the bootstrap procedure has another advantage that has been unrecognized by the academic literature thus far. This is the ability of bootstrap to preserve high accuracy despite using a smaller training set than test set. We believe that this is an important finding, as it can be applied in smaller datasets (where the test set is limited) as well as large datasets (where the execution time can be decreased).

## 5. Conclusion

In our research, we have shown that the bootstrap procedure can be applied as a substitute to the cross validation procedure in support vector machines. The many benefits of the tenfold bootstrap procedure include a significant reduction of computing time even in non-transformed data and reliable classification measures (which is comparable to those from the cross validation procedure). Our algorithm avoids overfitting while proving to be simple for application. The tenfold bootstrap procedure we propose produces results that are easy for interpretation, as we do not apply preliminary transformations on the data.

Another advantage is that the training/test split can be performed using smaller as well as larger test sets. This is particularly important in smaller datasets, where testing observations can be limited. With this finding, we propose a novel approach for reducing computational time when fitting support vector machines.

## Acknowledgements

*The research presented in this paper was supported by Sofia University's Research Program.*

## References


[1] Bootstrapped 0.0.2. https://pypi.org/project/bootstrapped/. Accessed: 2019–12–15.

[2] Barboza F., Kimura H., Altman E.: Machine Learning Models and Bankruptcy Prediction, *Expert Systems with Applications*, vol. 83, pp. 405–417, 2017.

[3] Berrar D.: Introduction to the Non-Parametric Bootstrap. In: *Encyclopedia of Bioinformatics and Computational Biology, Volume 1*, Elsevier, pp. 766–773, 2019.

[4] Breiman L.: The Little Bootstrap and Other Methods for Dimensionality Selection in Regression: X-fixed Prediction Error, *Journal of American Statistical Association*, vol. 87, pp. 738–754, 1994.

[5] Breiman L.: Better Subset Regression Using the Nonnegative Garrote, *Technometrics*, vol. 37, pp. 373–384, 1995.

[6] Chatzis S., Siakoulis V., Petropoulos A., Stavroulakis E., Vlachogiannakis N.: Forecasting stock market crisis events using deep and statistical machine learning techniques, *Expert Systems with Applications*, vol. 112, pp. 353–371, 2018.

[7] Cortes C., Vapnik V.: Support-vector networks, *Machine Learning*, vol. 20, pp. 273–297, 1995.

[8] Efron B.: Bootstrap Methods: Another Look at the Jackknife, *The Annals of Statistics*, vol. 7(1), pp. 1–26, 1979. https://www.jstor.org/stable/2958830

[9] Efron B., Tibshirani R.: Improvements on Cross-Validation: The .632+ Bootstrap Method, *Journal of the American Statistical Association*, vol. 92, pp. 548–560, 1997.

[10] Frunza M.C.: *Solving Modern Crime in Financial Markets*, Academic Press, Chapter 2I – Support Vector Machines, 2016.

[11] Hoerl A.E., Kennard R.W.: Ridge Regression. Applications to nonorthogonal problems, *Technometrics*, vol. 12(1), pp. 69–82, 1970.

[12] James G., Witten D., Hastie T., Tibshirani R.: *An Introduction to Statistical Learning*, Springer, 2013.

[13] Khairunnahar L., Hasib M.A., Rezanur R.H.B., Islam M.R., Hosain M.K.: Classification of malignant and benign tissue with logistic regression, *Informatics in Medicine Unlocked*, vol. 16, 2019.

[14] Krstajic D., Buturovic L.J., Leahy E., Thomas S.: Cross-validation pitfalls when selecting and assessing regression and classification models, *Journal of Cheminformatics*, vol. 6, p. 10, 2014.

[15] Luo X., Zhu X., Lim E.G.: A parametric bootstrap algorithm for cluster number determination of load pattern categorization, *Energy*, vol. 180, pp. 50–60, 2019.

[16] MacKinnon J.G.: Bootstrap Inference in Econometrics, *The Canadian Journal of Economics*, vol. 35, pp. 615–645, 2002.

[17] Maldonado S., Pérez J., Weber R., Labbé M.: Feature selection for Support Vector Machines via Mixed Integer Linear Programming, *Information Sciences*, vol. 279, pp. 163–175, 2014.

[18] Morais C.L.M., Lima K.M.G., Martin F.L.: Uncertainty estimation and misclassification probability for classification models based on discriminant analysis and support vector machines, *Analytica Chimica Acta*, vol. 1063, pp. 40–46, 2019.

[19] Olejnik S., Mills J., Keselman H.: Using Wherry's Adjusted $R^2$ and Mallow's $C_p$ for Model Selection from All Possible Regressions, *The Journal of Experimental Education*, vol. 68, pp. 365–380, 2000.

[20] Ozcan B., Ozturk I.: Renewable energy consumption-economic growth nexus in emerging countries: A bootstrap panel causality test, *Renewable and Sustainable Energy Reviews*, vol. 104, pp. 30–37, 2019.

[21] Tibshirani R.: Regression shrinkage and selection via the lasso: a retrospective, *Journal of the Royal Statistical Society Series B (Statistical Methodology)*, vol. 73(3), pp. 273–282, 2011.

[22] Vrigazova B.: Nonnegative Garrote as a Variable Selection Method in Panel Data, *International Journal of Computer Science and Information Security*, vol. 16(1), pp. 95–106, 2018.

[23] Vrigazova B., Ivanov I.: Optimization of the ANOVA Procedure for Support Vector Machines, *International Journal of Recent Technology and Engineering*, vol. 8(4), pp. 5160–5165, 2019.

[24] Vrigazova B., Ivanov I.: The bootstrap procedure in classification problems, *International Journal of Data Mining, Modelling and Management*, vol. 12, 2020, in press.

[25] Wong T.-T.: Performance evaluation of classification algorithms by $k$-fold and leave-one-out cross validation, *Pattern Recognition*, vol. 48(9), pp. 2839–2846, 2015.

[26] Xie Z., Chen S.: Exchange rates and fundamentals: A bootstrap panel data analysis, *Economic Modelling*, vol. 78, pp. 209–224, 2019.

[27] Zhang Y., Yang Y.: Cross-validation for selecting a model selection procedure, *Journal of Econometrics*, vol. 1, pp. 95–112, 2015.

[28] Zou H.: The Adaptive Lasso and Its Oracle Properties, *Journal of the American Statistical Association*, vol. 101, 2006.

[29] Zou N., Politis D.N.: Bootstrap seasonal unit root test under periodic variation, *Econometrics and Statistics*, 2020.

## Affiliations

**Borislava Vrigazova**
Sofia University, Bulgaria, vrigazova@uni-sofia.bg,
ORCID ID: https://orcid.org/0000-0001-9335-6927

**Ivan Ivanov**
Sofia University, Bulgaria, i_ivanov@feb.uni-sofia.bg

**Received**: 28.01.2020
**Revised**: 29.03.2020
**Accepted**: 03.04.2020