Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification

Sanjay Yadav
M.Tech. in Computer Networking
Maulana Azad National
Institute of Technology
Bhopal, Madhya Pradesh 462003
Email: yadavsanjay116@hotmail.com

Sanyam Shukla
Computer Science and Engineering
Maulana Azad National
Institute of Technology
Bhopal, Madhya Pradesh 462003
Email: sanyamshukla@gmail.com

***Abstract*—While training a model with data from a dataset, we have to think of an ideal way to do so. The training should be done in such a way that while the model has enough instances to train on, they should not over-fit the model and at the same time, it must be considered that if there are not enough instances to train on, the model would not be trained properly and would give poor results when used for testing. Accuracy is important when it comes to classification and one must always strive to achieve the highest accuracy, provided there is not trade off with inexcusable time. While working on small datasets, the ideal choices are k-fold cross-validation with large value of k (but smaller than number of instances) or leave-one-out cross-validation whereas while working on colossal datasets, the first thought is to use holdout validation, in general. This article studies the differences between the two validation schemes, analyzes the possibility of using k-fold cross-validation over hold-out validation even on large datasets. Experimentation was performed on four large datasets and results show that till a certain threshold, k-fold cross-validation with varying value of k with respect to number of instances can indeed be used over hold-out validation for quality classification.**

***Index Terms*—Cross-validation, k-fold cross-validation, hold-out validation, colossal datasets, classification.**

I. Introduction

Machine learning can simply be defined as using data instead of logic to perform tasks by a machine. We use data to train the machine, as in, tell it what it has to do and then test the trained model on different tasks to see whether the training has been successful or not. When it comes to data mining, the term classification plays an important role as it assigns class values to new instances found during the data mining.

In statistics or data mining, one of the main tasks is to train a model from available data. Such models may be regression models or classifier models. While such models may show adequate prediction capability on the training data, at the same time, they might also not predict the future data correctly. This is a problem that needs to be solved. Cross-validation (CV) [1]–[3] is a procedure for estimating the generalization performance in this context. Cross-validation is the most commonly used method for predictive performance evaluation of a model, given beforehand or when it is developed by a modeling procedure. Data is split usually into two parts and based on this splitting, on one part, training is done while the predictive performance is tested on the other part. The training-and-testing scheme works equally well for classification models of machine learning. We train a model using some instances of the dataset and leave some instances out of it to test the model after it has been trained. This is the underlying principle in cross-validation. Thus, cross-validation is widely accepted in data mining and machine learning community, and serves as a standard procedure for the sake of model selection or modeling procedure selection [4], [5].

The most critical factor in applying cross validation is choosing data splitting ratio for the training and testing part. This ratio is also known as validation size (n_v). The choice of this is particularly hard to make and even a little mistake in choosing the validation size may lead to over-fitting or under-fitting. In general, there are two types of framework based on whether the true model is within the candidate model or not. The first is parametric framework, when the true model lies within the candidate model and when it is the reverse, we call that model a non-parametric model. In the first framework, the leave-one-out is equivalent to Akaike Information Model (AIC) [6]. They are inconsistent in a way that the probability of choosing the true model does not converge to 1 as the sample size *n* goes to ∞, while Bayesian Information Criterion (BIC) [7] and delete-n_v cross-validation with $n_v/n \rightarrow 1$ (and $n - n_v \rightarrow \infty$) are consistent [8]–[10]. Whereas in non-parametric framework, leave-one-out CV and AIC lead to asymptotically optimal or rate optimal choice for regression function estimation, while BIC and delete-n_v CV with $n_v/n \rightarrow 1$ usually lose the asymptotic optimality [11]–[13]. Therefore, the choice of the validation size or the choice of an information criterion is dependent on whether

978-1-4673-8286-1/16 $31.00 © 2016 IEEE
DOI 10.1109/IACC.2016.25

the data are under a parametric or a nonparametric framework.

When prior information on the true model is not available, a random use of model selection criteria may result in poor performance [13], [14]. In this context, for the problem of choosing the most appropriate modeling or model selection procedure for the data at hand, cross-validation provides a general solution. The two possible goals in cross-validation can be mentioned as follows [15]:

1) To predict the performance of a learnt model from available data using an algorithm, i.e, to see the generalizability of an algorithm.
2) To compare performances of multiple algorithms on a data and thus, find the best algorithm suitable for that data.

To meet the above two goals, various methods are proposed:

Re-substitution Validation: Here, model training is done on all the available data and then testing is done on the same data. Thus, this suffers from over-fitting.

Hold-out Validation: Since over-fitting is an issue of re-substition validation, in hold-out validation, we split the data into two parts, training on one while testing on other with the trained model.

K-fold Cross-validation: Here, data is split into k-equal parts. Training of the model is done on k-1 parts and one part is left out for testing. The process does not end here. The above is repeated k times while changing the test part one-by-one until testing has been done on all the k parts. See section III, subsection A for detailed explanation.

Leave One Out Cross-validation: In this, the number of folds k equals the number of data instances available. It suffers from high-variance [16].

Repeated k-fold Cross-validation: To obtain reliable performance estimation or comparison, large number of estimates are always preferred. In k-fold cross-validation, the number of estimates obtained equal k. To achieve further increase in the number of estimates, we run k-fold cross-validation multiple times. The data is reshuffled and re-stratified before each round.

Generally speaking, the higher the value of k, the higher the accuracy in cross-validation. However, this may lead to over-fitting. Hence, leave-one-out cross-validation is only good for small datasets where the number of instances is generally less than 100. At the same time, when the data set is colossal, it is generally suggested that we use holdout validation to reduce the time that is taken while training the model. In this article, we aim to differ from the general unsaid suggestion of using the holdout method for colossal datasets and prove using results that the time tradeoff for using k-fold validation over holdout validation is worth the accuracy obtained while keeping the value of k as less as possible for quality classification. In this article, we see what are k-fold cross-validation, hold-out validation and the difference between hold-out validation with 50% hold-out and 2-fold cross-validation in section III, subsections A, B and C respectively. The novelty and motivation of this article comes from the fact that not many researchers focus on colossal datasets. See section II for related works.

II. Related Work

Recent advances made in the CV field are mentioned below.

A. *Tzu-Tsung Wong*

The above mentioned author wrote an article “Performance evaluation of classification algorithms by k-fold and leave-one-out cross-validation” [17]. In that article, the author has analyzed the usage of k-fold cross-validation and leave-one-out cross-validation. The author concluded that four factors, viz, number of folds, number of instances in the fold, level of averaging and repetition of cross-validation affect the outcome of cross-validation techniques. Sampling Distributions and Independence Assumptions were proposed to compare the two. The article, thus, gave us an idea of cross-validating techniques on small datasets and the factors that affect them. However, the idea was general and leave-on-out cross-validation technique can only be used on small datasets. Nothing was mentioned for colossal datasets.

B. *Yongli Zhang, Yuhong Yang*

The authors mentioned above wrote an article “Cross-validation for selecting a model selection procedure” [5]. This article teaches us about selecting a model for regression or classification. The article goes in-depth in order to understand cross-validation and its use in selecting an ideal model for a task that needs to be done. The authors proposed 2nd level cross-validation and a clear subsection has been written on this to analyze whether or not we need the second level cross-validation for selecting a model. The article contradicted the work of Kohavi [18] of using 10-fold CV in general. And we agree, 10-fold CV can not be used on every dataset, especially on colossal datasets. The number of folds highly impact the computational time of the classification algorithm. Since, number of folds is directly proportional to computational time, 10-fold CV on colossal datasets is not recommended.

C. *Payam Refaeilzadeh, Lei Tang, Huan Liu*

The article, “Cross-validation” [15], is written by the above mentioned authors. The article explains all the cross-validation techniques in brief, as well as their merits and de-merits. However, like others, this article also did not mention anything on colossal datasets. Even though the article written was independent of size of the dataset, the article did give us a general idea on all the cross-validation techniques. The article was completely a theoretical one, no experiments were performed.

III. Proposed Work

In related work, k-fold cross-validation benefits have been discussed. However, this article deals with analyzing the performance of k-fold cross-validation on colossal datasets. Most datasets do not have a lot of instances and hence, for this paper, we take the threshold at thousand instances. If the number of instances in the dataset is more than a thousand, we term that dataset as a colossal dataset. Let us see the main topics of this article in a little detailed manner.

A. K-fold Cross-validation

In this subsection, we will explain k-fold cross-validation in detail. Firstly, the data is equally partitioned into k equal or nearly equal segments or folds. On these partitioned folds, training and testing is done in k iterations such that in each iteration, we leave one fold for testing and train the model on the remaining k-1 folds. The accuracy obtained in each iteration is then averaged to get the model accuracy. An important thing to note is that data is commonly stratified before being split into k segments. Stratification is the process of rearranging data in such a way that each fold is a good representative of the whole. To understand this, consider a 4-class classification problem where every class consists of 25% of the data, it will be a good practice to arrange the data in such a way that each fold consists around quarter of the total instances.

B. Hold-out Validation

Hold-out validation was proposed to remove the problem of over-fitting that was there in re-substitution validation. Here, the data is divided into two non-overlapping parts and these two parts are used for training and testing respectively. The part which is used for testing is the hold-out part. It is so named because we hold-out that part for testing and learn the model using the remaining part of the data. People usually do not understand this very clearly and assume that hold-out validation is splitting the data into two equal parts. While its true that it can be called hold-out validation, however, it is a very specific case of hold-out validation where the amount of data being held-out for testing is 50%.

Thus, hold-out validation can have different percentages of data being held-out for testing. We can use 20% hold-out validation or even 10% hold-out validation but at the same time, 10% hold-out validation sometimes, may suffer from over-fitting if the 90% data being used for learning the model is not properly distributed and differs vastly from the 10% testing data. Note that in using hold-out validation, the time taken for learning the model is relatively lesser than the time taken for learning the model using k-fold cross-validation.

C. Difference between 2-fold CV and 50%hold-out validation

In 50% hold-out validation, data is split into two equal parts and on one part, the model is trained and on the other, the trained model is tested. This is done only once. However, in 2-fold cross-validation, even though the data is again split into two equal parts, the model is trained on each part in 2 different iterations, and the testing is done on the other part, respectively. The accuracy obtained in both the iteration is then averaged to get the mean accuracy of the model. This, relatively simple difference between the two, is wrongly understood by even some experts of machine learning and hence, there are many articles on the web claiming that both the validation techniques are same. In order to understand the basic working of the two validation techniques, one must be clear on this difference.

D. Proposed Point of Comparison

After the above difference has been understood, one must be able to visualize and interpret that even 10% hold-out validation and 10-fold cross-validation are similar, except in 10-fold cross validation, the model is trained and tested 10 different times and then, mean accuracy is considered. Similarly, 5-fold cross-validation and 20% hold-out validation are similar as they both train 80% of the data and test on the remaining 20%. Does 5-fold CV really train on 80% of the data? Yes. The data is split into 5-equal folds in 5-fold CV and hence in each fold, 20% of the data is available. One fold is left for testing and the remaining four folds are used for training. In each fold 20% of the data is available and hence in four folds, 20 X 4 = 80% of the data is available. Clearly, the difference between the two is that in 5-fold CV, the model is trained and tested for 5 times and then, the mean accuracy is considered as the accuracy of the model. It must be noted that even though percentage of withholding the data for testing can be taken as a point of comparison, the data actually being held out is different in every iteration, i.e., it is random in k-fold cross-validation and hence the accuracy may vary in every run. However, the accuracy change isn't much in every run and hence, we have taken the percentage of data holding for testing as our point of comparison.

IV. Experimental Results

We used some of the popular datasets available on the UCI data repository and obtained the results of k-fold cross-validation as well as hold-out validation on them. The datasets have the following properties:

- Number of instances: > 1000.
- Number of attributes: 10-100.
- Type : Multivariate.
- Under: Classification.

The experiments were performed on Intel core i7 dual core processor with 4GB RAM. The physical memory limit posed a problem on performing the classification on colossal datasets as "Not having enough memory was returned" upon running the algorithms. Hence, the datasets on which the experiments were performed have instances between 1000 and 50,000.

We used 20 algorithms on each data set. These algorithms, with one-line details, are as follows:

- Decision Trees
  - *Complex Tree*
    - A decision tree with many leaves that makes many fine distinction between classes. Upto 100 splits.
  - *Medium Tree*
    - Medium-sized decision tree with relatively fewer number of splits. Upto 20 splits.
  - *Simple Tree*
    - A simple decision tree that makes coarse decisions between classes. Upto 4 splits.

- Support Vector Machine
  - *Linear SVM*
    - A simple SVM that makes linear seperation between classes, using the linear kernel.
  - *Quadratic SVM*
    - This SVM is based on quadratic kernel.
  - *Cubic SVM*
    - This SVM makes use of the cubic kernel.
  - *Fine Gaussian*
    - A SVM that makes fine distinction between classes with the help of the Gaussian kernel with kernel scale set to $\sqrt{P}/4$ where P is number of predictors.
  - *Medium Gaussian*
    - A SVM that makes relatively fewer distinction between classes with the help of the Gaussian kernel with kernel scale set to $\sqrt{P}$ where P is number of predictors.
  - *Coarse Gaussian*
    - A SVM that makes coarse distinction between classes with the help of the Gaussian kernel with kernel scale set to $\sqrt{P}*4$ where P is number of predictors.

- K-Nearest Neighbor
  - *Fine KNN*
    - A nearest-neighbor classifier that makes finely detailed distinctions between classes with number of neighbors set to 1.
  - *Medium KNN*
    - A nearest-neighbor classifier that makes fewer distinctions between classes than a fine KNN, with number of neighbors set to 10.
  - *Coarse KNN*
    - A nearest-neighbor classifier that makes finely detailed distinctions between classes with number of neighbors set to 100.
  - *Cosine KNN*
    - A nearest-neighbor classifier that makes use of cosine distance metric.
  - *Cubic KNN*
    - A nearest-neighbor classifier that makes use of cubic distance metric.
  - *Weigthed KNN*
    - A nearest-neighbor classifier that makes use of weighted distances.

- Ensembles
  - *Bagged Ensemble*
    - This model creates an ensemble of simple decision trees using the Adaboost algorithm. Compared to Bagging, this model may need relatively less time or memory but require more ensemble members.
  - *Boosted Ensemble*
    - A bootstrap-aggregated ensemble of complex decision trees. Often very accurate, but can be slow and require large memory for large datasets.
  - *Subspace Discriminant(SD)*
    - This ensemble is good for many predictors. Fast classification and less memory usage but the accuracy varies, depending on the data.
  - *Subspace KNN*
    - This classifier is used when there are many predictors. The model creates an ensemble of nearest-neighbor classifiers using the Random Subspace method.
  - *RUSBoost*
    - This classifier is mainly used when the data is skewed with many more observations of one class. Thus, this classifier will give much poor accuracy with our datasets. However, even with poor accuracy, we can still compare hold-out validation and k-fold cross-validation.

The results of these classification algorithms on some of the colossal datasets are shown below in tabular format. Results on Page Blocks Dataset, Optical Recognition of Handwritten Digits Dataset, Magic Gamma Telescope Dataset and Letter Recognition Dataset are shown in table I, II, III and IV respectively. A summary on the results of these datasets is shown in table V.

Since we are analyzing the performance of k-fold cross-validation over hold-out validation, for cases when the accuracy returned by both the models is same, we have highlighted the case in favor of k-fold cross-validation in order to see the percentage of cases where k-fold cross-validation prevails over hold-out validation. In table V, A is number of cases when k-fold cross-validation gives better or equal accuracy than hold-out validation. B is number of

TABLE I
PAGE BLOCKS CLASSIFICATION

Algorithm	*50% Hold-out*	*2-fold CV*	*20% Hold-out*	*5-fold CV*
Complex Tree	96.3	**96.3**	96.3	**96.4**
Medium Tree	96.7	**96.7**	96.2	**96.6**
Simple Tree	**95.9**	95.7	94.3	**95.7**
Linear SVM	**96.4**	96.1	96.1	**96.2**
Quadratic SVM	**96.9**	96.7	96.9	**96.9**
Cubic SVM	96.3	**96.6**	96.5	**96.9**
Fine Gaussian	94.4	**94.9**	94.7	**95.5**
Medium Gaussian	95.2	**96.1**	95.9	**96.5**
Coarse Gaussian	93.3	**94.2**	94.2	**94.8**
Fine KNN	95.8	**96.1**	95.8	**96.3**
Medium KNN	95.2	**95.9**	95.3	**96.1**
Coarse KNN	92.1	**92.5**	92.9	**93.4**
Cosine KNN	95.1	**95.7**	95.2	**95.8**
Cubic KNN	95.2	**95.8**	95.3	**96.0**
Weighted KNN	96.5	**96.6**	96.0	**96.9**
Boosted Ensemble	92.7	**93.2**	92.9	**93.1**
Bagged Ensemble	**97.6**	97.3	97.0	**97.5**
SD	92.4	**92.9**	92.5	**92.7**
Subspace KNN	95.4	**95.6**	94.9	**95.9**
RUSBoost	7.4	**37.5**	7.2	**26.7**

TABLE II
OPTICAL RECOGNITION OF HANDWRITTEN DIGITS

Algorithm	*50% Hold-out*	*2-fold CV*	*20% Hold-out*	*5-fold CV*
Complex Tree	87.1	**87.3**	87.7	**88.5**
Medium Tree	**70.3**	69.6	70.8	**70.9**
Simple Tree	38.4	**39.1**	38.4	**40.0**
Linear SVM	96.9	**97.4**	97.1	**97.5**
Quadratic SVM	98.1	**98.4**	98.0	**98.6**
Cubic SVM	98.5	**98.6**	**98.2**	98.0
Fine Gaussian	67.2	**67.6**	69.2	**70.0**
Medium Gaussian	97.3	**97.6**	97.2	**97.9**
Coarse Gaussian	94.6	**95.3**	95.1	**95.8**
Fine KNN	97.5	**97.7**	97.8	**97.8**
Medium KNN	96.9	**96.9**	96.5	**97.2**
Coarse KNN	92.0	**92.5**	**94.5**	93.8
Cosine KNN	95.8	**96.0**	96.6	**96.7**
Cubic KNN	**96.0**	95.9	**97.0**	96.5
Weighted KNN	97.4	**97.4**	97.0	**97.6**
Boosted Ensemble	**40.8**	39.7	**42.1**	38.1
Bagged Ensemble	97.3	**97.8**	98.0	98.0
Subspace Discriminant	94.5	**94.9**	95.0	**95.3**
Subspace KNN	98.4	**98.7**	98.7	**98.8**
RUSBoost	19.0	**19.4**	19.1	**21.4**

cases when hold-out validation gives better accuracy than k-fold cross-validation. In all the four datasets, the number of cases equal 40. 20 algorithms applied once on 2-fold CV and 50% hold-out and then applied again on 5-fold and 20% hold-out. C is percentage of A over the 40 cases and D is percentage of B over the 40 cases on the datasets respectively. The percentages are calculated to see the effectiveness in percentage values for better analyzing the performance of k-fold CV over hold-out validation.

TABLE III
MAGIC GAMMA TELESCOPE

Algorithm	*50% Hold-out*	*2-fold CV*	*20% Hold-out*	*5-fold CV*
Complex Tree	84.2	**84.2**	85.2	**85.6**
Medium Tree	82.6	**83.0**	82.3	**84.0**
Simple Tree	79.1	**79.4**	**79.3**	79.2
Linear SVM	**79.7**	79.2	79.1	**79.2**
Quadratic SVM	86.0	**86.1**	86.4	**87.1**
Cubic SVM	86.2	**86.9**	86.5	**86.9**
Fine Gaussian	85.5	**85.6**	**86.2**	86.1
Medium Gaussian	**87.2**	86.6	**86.4**	86.0
Coarse Gaussian	82.2	**82.6**	83.5	**83.7**
Fine KNN	80.7	**80.9**	81.6	**82.3**
Medium KNN	**83.0**	82.9	83.3	**83.5**
Coarse KNN	81.0	**81.0**	82.2	**82.2**
Cosine KNN	83.1	**83.7**	84.0	**84.2**
Cubic KNN	82.4	**82.9**	83.0	**83.5**
Weighted KNN	83.1	**84.0**	84.3	**85.1**
Boosted Ensemble	**82.0**	81.4	**81.9**	81.3
Bagged Ensemble	87.6	**87.8**	88.1	**88.5**
Subspace Discriminant	77.0	**77.6**	76.6	**77.8**
Subspace KNN	**82.6**	82.2	**83.0**	82.6
RUSBoost	71.7	**72.5**	73.7	**74.0**

TABLE IV
LETTER RECOGNITION

Algorithm	*50% Hold-out*	*2-fold CV*	*20% Hold-out*	*5-fold CV*
Complex Tree	**62.6**	62.4	61.2	**62.0**
Medium Tree	**37.0**	36.4	35.7	**36.3**
Simple Tree	15.5	**16.0**	15.9	**15.9**
Linear SVM	83.8	**83.9**	84.4	**84.7**
Quadratic SVM	94.1	**94.4**	95.4	**95.7**
Cubic SVM	95.0	**95.4**	96.5	**96.6**
Fine Gaussian	**89.4**	88.0	91.1	**91.5**
Medium Gaussian	93.0	**93.1**	94.5	**94.8**
Coarse Gaussian	**79.4**	79.1	80.6	**81.5**
Fine KNN	93.7	**93.8**	94.8	**95.2**
Medium KNN	91.5	**91.8**	93.5	**93.7**
Coarse KNN	77.7	**77.9**	**82.8**	82.6
Cosine KNN	**90.5**	90.2	**92.6**	92.4
Cubic KNN	90.1	**90.4**	92.4	**92.6**
Weighted KNN	93.5	**93.7**	95.0	**95.0**
Bagged Ensemble	95.0	**95.4**	96.6	**96.6**
Boosted Ensemble	**13.1**	10.3	10.3	**10.9**
RUSBoost	**7.0**	6.9	6.8	**6.9**
Subspace Discriminant	66.7	**67.4**	66.4	**66.9**
Subspace KNN	95.2	**95.3**	96.2	**96.3**

TABLE V
PERFORMANCE OF K-FOLD CV OVER HOLD-OUT VALIDATION

Dataset	*#Ins.*	*A*	*B*	*C*	*D*
Page Blocks Classification	5473	36	4	90	10
Optical Recognition of Handwritten Digits	5620	32	8	80	20
Magic Gamma Telescope Dataset	19020	30	10	75	25
Letter Recognition Dataset	20000	31	9	77.5	22.5

V. Conclusion

After analyzing the usage of k-fold cross-validation over hold-out validation on colossal datasets and as per the result summary in table 5, we can see that the percentage of C is much higher than that of D, indicating that we can indeed go for k-fold cross-validation over hold-out validation on colossal datasets for quality classification. We would like to conclude by mentioning the following points:

- We can definitely use k-fold cross-validation over hold-out validation even after the time trade-off because using the k-fold cross-validation gives us around 0.1-3% more accurate result, in general, as opposed to the hold-out validation over the same classification algorithm.
- However, the time trade-off for colossal datasets can be huge and hence, a reality check must be there to limit the value of k in k-fold cross-validation.
- With the experiments performed, we can predict the value of k for which even the time trade-off will be worth it. We were limited by the physical memory to test it, however, the results obtained on slightly smaller datasets point out towards the values given in the table below. A further deep experimentation needs to be performed to verify the same.

Number of Instances	***Value of k***
5000-100000	5-6
100000-1000000	3-5
>1000000	2-3

References

[1] D. M. Allen, "The relationship between variable selection and data agumentation and a method for prediction," *Technometrics*, vol. 16, no. 1, pp. 125–127, 1974.

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

[3] S. Geisser, "The predictive sample reuse method with applications," *Journal of the American Statistical Association*, vol. 70, no. 350, pp. 320–328, 1975.

[4] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: data mining, inference and prediction," *The Mathematical Intelligencer*, vol. 27, no. 2, pp. 83–85, 2005.

[5] Y. Zhang and Y. Yang, "Cross-validation for selecting a model selection procedure," *Journal of Econometrics*, vol. 187, no. 1, pp. 95–112, 2015.

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

[7] G. Schwarz *et al.*, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[8] M. Stone, "Asymptotics for and against cross-validation," *Biometrika*, pp. 29–35, 1977.

[9] R. Nishii *et al.*, "Asymptotic properties of criteria for selection of variables in multiple regression," *The Annals of Statistics*, vol. 12, no. 2, pp. 758–765, 1984.

[10] J. Shao, "Linear model selection by cross-validation," *Journal of the American statistical Association*, vol. 88, no. 422, pp. 486–494, 1993.

[11] K.-C. Li, "Asymptotic optimality for cp, cl, cross-validation and generalized cross-validation: discrete index set," *The Annals of Statistics*, pp. 958–975, 1987.

[12] T. Speed and B. Yu, "Model selection and prediction: normal regression," *Annals of the institute of statistical mathematics*, vol. 45, no. 1, pp. 35–54, 1993.

[13] J. Shao, "An asymptotic theory for linear model selection," *Statistica Sinica*, vol. 7, no. 2, pp. 221–242, 1997.

[14] Y. Yang, "Prediction/estimation with simple linear models: is it really that simple?" *Econometric Theory*, vol. 23, no. 01, pp. 1–36, 2007.

[15] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross validation, encyclopedia of database systems." 2009.

[16] B. Efron, "Estimating the error rate of a prediction rule: improvement on cross-validation," *Journal of the American Statistical Association*, vol. 78, no. 382, pp. 316–331, 1983.

[17] T.-T. Wong, "Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation," *Pattern Recognition*, vol. 48, no. 9, pp. 2839–2846, 2015.

[18] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2, 1995, pp. 1137–1145.