Finding Unexpected Test Accuracy by Cross Validation in Machine Learning

Hoijin Yoon

Hyupsung University, Gyeonggi-do, Republic of Korea

Summary
Machine Learning(ML) splits data into 3 parts, which are usually 60% for training, 20% for validation, and 20% for testing. It just splits quantitatively instead of selecting each set of data by a criterion, which is very important concept for the adequacy of test data. ML measures a model's accuracy by applying a set of validation data, and revises the model until the validation accuracy reaches on a certain level. After the validation process, the complete model is tested with the set of test data, which are not seen by the model yet. If the set of test data covers the model's attributes well, the test accuracy will be close to the validation accuracy of the model. To make sure that ML's set of test data works adequately, we design an experiment and see if the test accuracy of model is always close to its validation adequacy as expected. The experiment builds 100 different SVM models for each of six data sets published in UCI ML repository. From the test accuracy and its validation accuracy of 600 cases, we find some unexpected cases, where the test accuracy is very different from its validation accuracy. Consequently, it is not always true that ML's set of test data is adequate to assure a model's quality.
Key words:
Software Testing, Machin learning, Test adequacy, Validation, Accuracy.

1. Introduction

In recent years, Machine Learning (ML) has become an essential technique for developing an intelligent system, which is built based on a big size of data. A data set is divided by 3 separate groups, which are one for training, another for validation, and the other for testing. For the convenience, these three part are called simply *Training set*, *Validation set*, and *Test set* as shown in Figure 1. As shown in Figure 1, the validation set is replaced with cross validation when the size of data is not big enough to save another group of data for validation. Cross validation generates a set of data for validation by a method, for example, n-folding and so on.

Training set is used to train a model, and the trained model understands the training data well. *Training accuracy*, the hit ratio of the model's predictions for a set of training data, must be high. But the fact that a model achieves a high training doesn't mean that it's a good model for all the general data. This is "Overfitting" problem. To avoid overfitting problem, the model should be validated. The data set for validation is applied to a model to measure *Validation accuracy*, which is a percentage of how many predictions hit their labels of data. The fact that the validation accuracy is much lower than the training accuracy means that the current model must be overfitting. Overfitting is a phenomenon often seen when a trained model performs extremely well on the samples used for training but performs poorly on new unknown samples; that is to say the model is not generalized well [1]. So the model needs to be generalized more by fixing the hyper-parameters [2]. The generalization has repeated until the model's validation accuracy is measured as high as its training accuracy. The generalized model is tested with test set, which has not been seen by the model yet. *Test accuracy* is a percentage of correct predictions for these unseen data. Figure 1 explains how 3 types of data sets are used in training, validating, and testing in ML, and introduces ML's terminology.

ML splits the data into 3 parts, which is usually 60% for training, 20% for validation, and 20% for testing instead of selecting data by a certain criterion. This approach is quantitative, and we doubt the quality of test set with wondering if the test set made by data-splitting could cover all the factors that should be tested in a model. Some

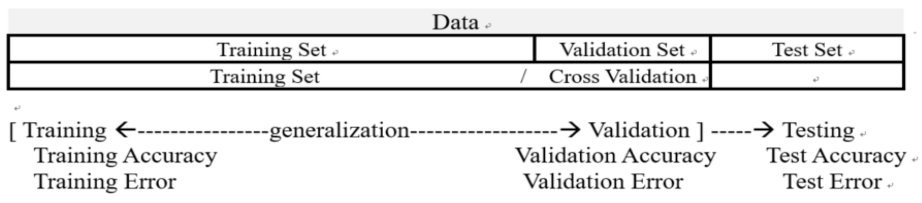


Fig 1. Training, Validation, and Testing in ML

Manuscript received December 5, 2021
Manuscript revised December 20, 2021
https://doi.org/10.22937/IJCSNS.2021.21.12.76

existing research works [3-6] also studies ML's data usage. Technically, the validation set is neutral in order to check if a model is overfitting, and test accuracy is expected to be close to the validation accuracy. If the test accuracy is still high as expected by a high validation accuracy, the test set can be said to be adequate and we don't have any reason for asking ML a certain criterion to select a test set. This research work answers the question about quality of ML's test set through an experiment checking if the test set covers the expectation that test accuracy is close to validation accuracy within a certain range.

Section 2 introduces two resources for the experiment; Scikit-learns and UCI repository. In Section 3, an experiment is designed and the charts resulted from the experiment is presented. Section 4 makes a discussion on some unexpected findings to answer our research question. The conclusion is mentioned in Section 5.

2. Materials and methods

2.1 Scikit-Learn

Scikit-Learn is a project that was started in 2007 as a Google Summer of Code project by David Cournapeau. In 2010 the team made the first public release. Scikit-learn is a Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. This package focuses on bringing machine learning to non-specialists using a general-purpose high-level language. Emphasis is put on ease of use, performance, documentation, and API consistency. It has minimal dependencies and is distributed under the simplified BSD license, encouraging its use in both academic and commercial settings. Source code, binaries, and documentation can be downloaded from http://scikit-learn.sourceforge.net [7].

Scikit-learn can evaluate a model's performance using cross-validation, and it maximizes computational efficiency as shown in Table 2[7], which compares computation time for some algorithms, implemented in the major machine learn toolkits accessible in Python; MLpy[8], PyBrain[9], pymvpa[10], MDP[11], and Shogun[12]. The computation is with Madelon data set[13], 4400 instances and 500 features.

2.2 UCI ML repository

The well-established University of California Irvine (UCI) ML Repository (https://archive.ics.uci.edu/ml/) is a collection of databases, domain theories, and data generators that are used for the empirical analysis of ML algorithms. The archive has been cited over 1000 times, making it one of the top 100 most cited papers in computer science [14].

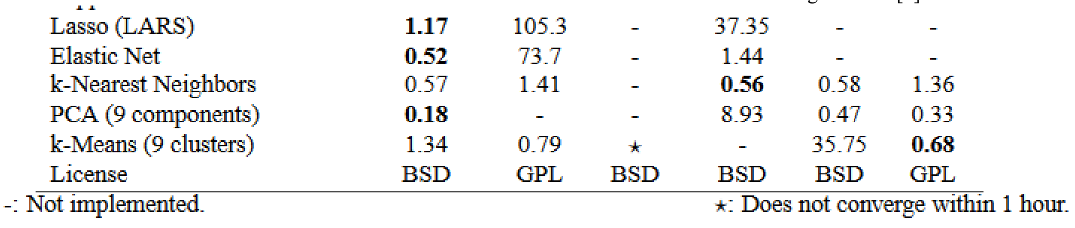
The Iris Flower dataset or Fisher's Iris dataset was introduced by the British statistician and biologist Ronald Fisher[15]. The dataset consists of 50 samples from each of the three species of Iris (Iris setosa, Iris virginica, and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other. For instance, the 35th sample in the Iris dataset is < 4: 9; 3: 1; 1: 5; 0: 2;" Irissetosa" >, where the sepal length is 4.9 cm, sepal width is 3.1 cm, petal length is 1.5 cm, petal width is 0.2 cm, and its class is Iris setosa. The dataset, which contains a set of 150 records, has been referred to in many research studies [16-20]. The experiment in this study uses the Iris Flower dataset obtained from a public research data storage, namely the Machine Learning Repository [14] maintained by the University of California, Irvine.

3. Results

3.1 Research Question

In ML, test set measures the accuracy of model that is passed through model validation. The model after the validation is expected to achieve a high accuracy under test set, which is a set of data unseen by a model yet. We doubt this expectation. That is why ML selects a test set quantitatively without a criterion while a traditional software testing selects test cases with a well-developed criterion. We want to make sure that ML's test set is good

Table 1. Time in seconds on the Madelon data set for various machine learning libraries [7]



Lasso (LARS)	**1.17**	105.3	-	37.35	-	-
Elastic Net	**0.52**	73.7	-	1.44	-	-
k-Nearest Neighbors	0.57	1.41	-	**0.56**	0.58	1.36
PCA (9 components)	**0.18**	-	-	8.93	0.47	0.33
k-Means (9 clusters)	1.34	0.79	⋆	-	35.75	**0.68**
License	BSD	GPL	BSD	BSD	BSD	GPL

-: Not implemented. ⋆: Does not converge within 1 hour.

enough for testing a validated model, which means that the test accuracy keeps close to its validation accuracy.

3.2. Experiment

The experiment aims to measure validation accuracy and test accuracy, and calculates the gap between validation accuracy and test accuracy. Technically, a high validation accuracy expects a high test accuracy and vice versa. The experiment runs 100 cases for each kind of data sets, and check if the test accuracy follows its validation accuracy. If a test accuracy is low with a high validation accuracy, test set or validation set is not selected adequately. Through the experiment, we find he unexpected test accuracy, and show that the ML's data splitting method needs to utilize data selection criteria that is considered in software testing.

3.2.1. Subjects

In UCI ML repository, "Most Popular Data Sets" are ranked based on the number of downloads since 2007. The top six data sets in this repository are applied to this experiment. The selected data sets have been used in many studies [15-17]. As described in Table 2, the six datasets are of different sizes from Iris (data volume: 150) to Adult (data volume: 48,842). Additionally, the feature complexity in each data set varies from simple Iris with 4 features to Breast Cancer with 32 features.

Table 2. The six most-popular datasets in the UCI machine learning repository

No.	Dataset	Size	Feature
1	Iris	150	4
2	Heart Disease	303	13
3	Breast Cancer	569	32
4	Car Evaluation	1728	6
5	Wine Quality	4898	12
6	Adults	48842	14

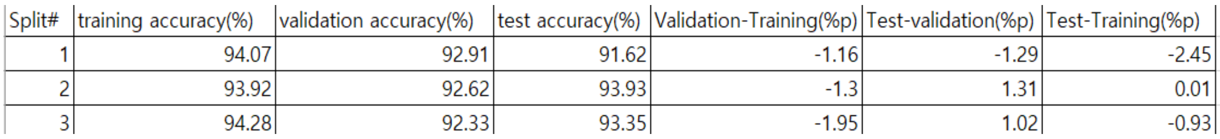
3.2.2. Measurement

For measurement, we coded a python program with Scikit-learn library. The program is executing the steps of experiment; splitting data into train set and test set, training a SVM model with the train set, validating it with cross validation set, measuring validation accuracy, testing the model with test set, measuring test accuracy, and calculating the gap between validation accuracy and test accuracy. It includes an iteration of 100 different data-splitting for each data set in it. The program generates validation set by n-folding method, where n equals to 5 for instance. That is because the size of some data sets in the experiment is not big enough to be divided into 3 parts; train set, validation set, and test set. In this case, cross validation is recommended traditionally. For example, 5-fold cross validation folds a train set into 5 parts and validates the model with each part of them. Finally, it calculates the average of five numbers of accuracy as the validation accuracy. We use n-fold cross validation in the experiment because cross validation is free from the potential overfitting problem caused by splitting quantitatively.

The output of experiment is written in a CSV file for each data set. The output includes 3 kinds of accuracy; train accuracy, validation accuracy, and test accuracy. It also includes the calculated gap between test accuracy and validation accuracy. Vertically, the file has 100 rows excepting the title row, and one row represents the output for one case of data splitting. Figure 2 is a part of the CSV file.

3.3. Validation Accuracy

As explained in Figure 1, a model is learning with *train set*, and then it is validated with *validation set*. At this time, the model is called as a hypothesis, since it is not a final version of model. The hypothesis has accepted data of *train set* in it through a learning process, but it hasn't seen data of validation set yet. Validation set as unseen data is very meaningful for the model selection, where it checks if the hypothesis works generally without overfitting to train set. Therefore, the validation set should be neural and moreover represents a future testing data. The experiment applies a cross validation method to avoid overfitting of validation data itself. Cross validation generates a set of data by folding data of train set. We use 5-fold method in the experiment.



Split#	training accuracy(%)	validation accuracy(%)	test accuracy(%)	Validation-Training(%p)	Test-validation(%p)	Test-Training(%p)
1	94.07	92.91	91.62	-1.16	-1.29	-2.45
2	93.92	92.62	93.93	-1.3	1.31	0.01
3	94.28	92.33	93.35	-1.95	1.02	-0.93

Fig 2 A part of output file for IRIS data set

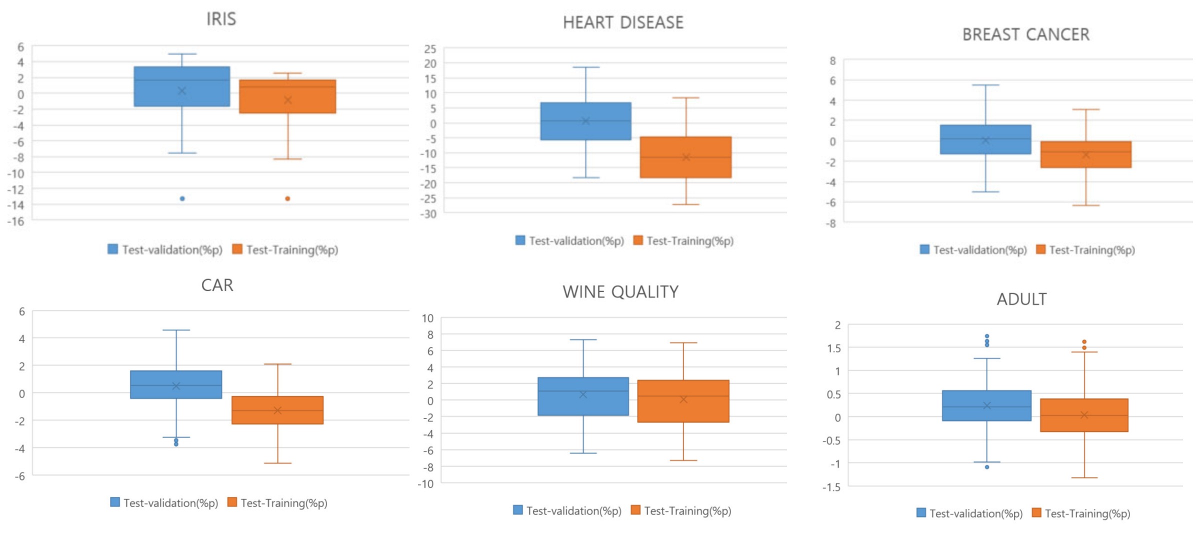


Fig 3. Box plot for the gap of Test accuracy from Validation accuracy or Train accuracy

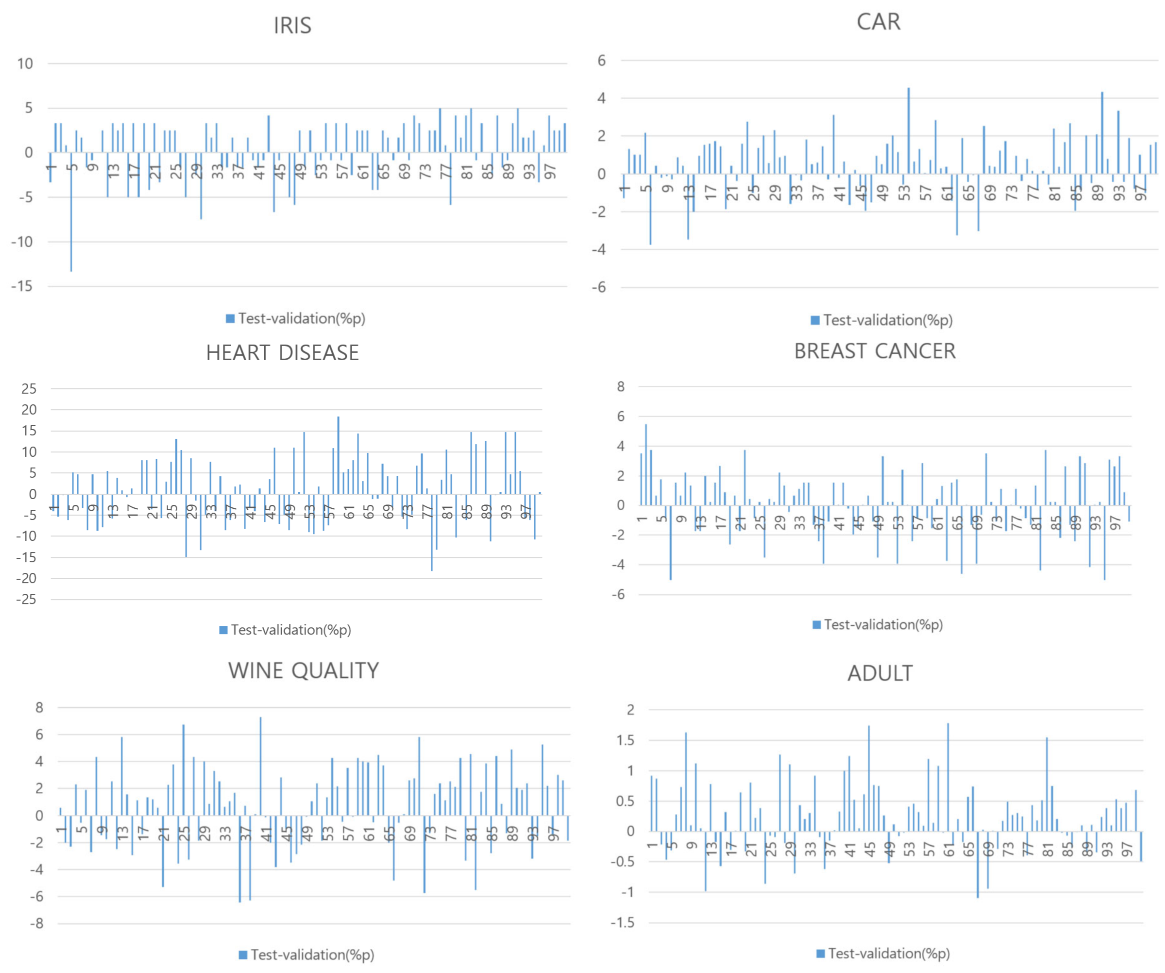


Fig 4. Bar chart representing the gap between validation accuracy and test accuracy

The result shows that the validation accuracy is always more close to its test accuracy compared to the train accuracy. In the chart, one box plot presents all the numbers of accuracy for 100 different kinds of data splitting. This measurement is done for each of six UCI data sets mentioned in Table 2.

Figure 3 shows six box plot chars, one for each data set. The perfect result is located in zero, which means that the validation or train accuracy is the same as test accuracy. A box representing "test accuracy minus validation accuracy" is located around zero more than a box representing "test accuracy minus train accuracy." As shown in all the chars of Figure 3, we have recognized that the validation accuracy is more similar to test accuracy. It means that the validation set, which is generated by 5 folding cross validation, is more neutral than the train set.

3.4. Test Accuracy

According to the definition of validation, a high validation accuracy expects a high test accuracy. The test set would be biased or not adequate for testing if the expectation does not always happen. As mentioned in Section 1, we focus on the adequacy of ML's test set in terms of software testing. The experiment calculates the gap between validation accuracy and test accuracy. The gap is calculated by a formula, test accuracy minus validation accuracy. The gap would be positive if test accuracy is higher, and negative if test accuracy is lower than validation accuracy. The length of line in chart of Figure 4 presents the gap between test accuracy and validation accuracy. The length would be short if test accuracy is almost same to the validation accuracy, and long if test accuracy is far from the validation accuracy.

In case of a long line in chart, the model passed though validation would not get a good test accuracy although it is supposed to achieve a high accuracy. This is the unexpected result, which exist in some cases of 100 different data splitting. For instance, the test accuracy of the 5th case in IRIS is almost 13%p lower than its validation accuracy. It must be a big difference.

4. Discussion

4.1 Unexpected test accuracy

According to the definition of validation in model selection, validation accuracy represent how much the model is accurate in predicting for general input data. Low validation accuracy means that the trained model is overfitting, and the model needs to be generalized in order to get a high validation accuracy. Model selection process prefers a high validation accuracy expecting that the test accuracy is close to its validation accuracy. Through the experiment, we found that this expectation is not always true. The unexpected finding is high test accuracy for low validation accuracy or low test accuracy for high validation accuracy. The former case is presented as a dropping line under x-axis of chart in Figure 4, while the latter case is presented as a rising line over x-axis.

For instance, the difference between test accuracy and validation accuracy is -13.33%p in the 5th model of IRIS data set. It has been represented as a long dropping line in a IRIS chart of Figure 4. In this case, test accuracy is measured as 86.67% even though validation accuracy is 100%. For another instance, the 59th case of Heart Disease data set is presented as a long rising line in a chart of Figure 4. It achieves low validation accuracy, 54.86%, but its test accuracy is measured as 73.33%. 73.33% is located within the 5 top high test accuracy in 100 cases of Heart Disease.

Table 3. A few of unexpected figures measured in the experiment

Data Set Name	IRIS	Heart Disease
Split#	5	59
Train accuracy(%)	100	64.98
Validation accuracy(%)	100	54.86
Test accuracy(%)	86.67	73.33
Test accuracy –validation accuracy (%p)	-13.33	18.47

4.2 Adequacy of test set

In traditional software testing, a set of test data has to be selected by an appropriate criterion because an exhaustive testing, that covers all available input data, is impossible theoretically. The criterion affects test coverage finally. ML also selects a set of test data, but it doesn't use a criterion. It only splits data set quantitatively; usually 60% for train set, 20% for validation set, and the other 20% for test set. Considering the importance of criteria in software testing, we doubt whether the test set would work adequately or not. This is called as the test adequacy.

As described in Section 4.1, some unexpected test accuracy measured by test set were found in the experiment. Graphically, the long dropping or rising lines represents the unexpected results. In case of Heart Disease, the length of line in chart is long either over or below x-axis. Therefore, test set in ML is not always adequate for testing the model.

4.3 Threats of validity

A threat to internal validity, is that the experiment generate a validation set through only 5-fold cross validation method. It can be 3-fold or 4-fold instead of 5-fold. But the number of foldings may not be critical for the validation accuracy comes as an average of all the validation accuracy calculated by each fold. One more threat is that the experiment doesn't cover the case of normal validation set, 20% of data set. In our defense, the normal validation set could be biased because of data-splitting's randomness.

A threat to external validity is that the experiment uses SVM classifier as a target model. If another modeling algorithm is used in the experiment, the figures would be different. But this research work doesn't focus on analysis which modeling algorithm matches to the expectation. The difference between modeling algorithms is not the interest in this work. Therefore, we have chosen SVM classifier, which offers very high accuracy compared to other classifiers such as logistic regression, and decision trees. SVM has been widely used in different fields and it can obtain high performance in many real world applications such as image retrieval [19], cancer recognition [20], text classification [21] and credit scoring [22].

5. Conclusions

The traditional software testing selects a set of test data by an appropriate criterion considering the theory that an exhaustive testing, that covers all available input data, is impossible. ML also selects a set of test data, but it only splits data set quantitatively; usually 60% for train set, 20% for validation set, and the other 20% for test set. Namely, it doesn't use a test data selection criterion. To make sure that ML's set of test data works adequately, we designed an experiment that shows the adequacy of ML's test set. The experiment used six data sets presented in Table 1, implemented SVM classifier as a target model, and includes scikit-learn library. The adequacy was analyzed based on the expectation that test accuracy keeps close to its validation accuracy within a certain range.

Through the experiment, we found that this expectation is not always true. The unexpected finding is high test accuracy for low validation accuracy or low test accuracy for high validation accuracy. There happens the unexpected test accuracy in all the cases of six data sets, and some of them showed big difference, such as 13.3%p or 18.84%p, from the expected accuracy. The experiment is done for 100 cases of data-splittings, some of them satisfied the expectation, but some of them didn't. Therefore, we found that the cause of difference falls on data of test set. It means that the test set of ML is not adequate for a reliable testing. We suggest that ML apply its appropriate criteria to select a test set, since even safety-critical system has been accepting the ML approach recently. Current data-splitting method is not adequate for validating future ML systems.

Acknowledgments

This work was supported by the Hyupsung University Research Grant of 2020.

References

[1] Yun Xu and Royston Goodacre. *On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning*. Journal of Analysis and Testing. 2:249-262. (2018)

[2] Andrew Ng. Model Selection and Train/Validation/Test sets. Machine Learning @ Coursera.

[3] Shin Nakajima and Kai Ngoc BUI. *Dataset Coverage for Testing Machine Learning Computer Programs*. Proceedings of 23rd Asia- Pacific Software Engineering Conference; 2016 Dec 6-9; New Zealand: IEEE, (2016)

[4] Arnab Sharma and Keike Wehrheim. *Testing Machine Learning Algorithms for Balanced Data Usage.* Proceeding of 12th International Conference of Software Testing, Verification and Validation; 2019 April 22-27; China: IEEE; (2019)

[5] Senthil Mani and Anush Sankaran. *Coverage Testing of Deep Learning Models using Dataset Characterization*. ArXiv. 2019; arXiv:1911.07309.(2019)

[6] Du Zhang and Jeffrey Tsai. Machine Learning Applications in Software Engineering. World Scientific; (2005)

[7] F. Pedregosa et al. Scikit-learn: Machine Learning Systems with Python. Journal of Machine Learning Research. 12(85):2825-2830. (2011)

[8] D. Albanese, G. Merler, S.and Jurman, and R. Visintainer. MLPy: high-performance python package for predictive modelling. Proceeding on Workshop on Machine Learning Open Source Software. 2008 December 12; Canada: PASCAL. (2008)

[9] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. R¨uckstieß, and J. Schmidhuber. PyBrain. The Journal of Machine Learning Research. 11:743–746. (2010)

[10] M. Hanke, Y.O. Halchenko, P.B. Sederberg, S.J. Hanson, J.V. Haxby, and S. Pollmann. PyMVPA: A Python toolbox for multivariate pattern analysis of fMRI data. Neuro informatics. 7(1):37-53. (2009)

[11] T. Zito, N.Wilbert, L.Wiskott, and P. Berkes. Modular toolkit for data processing (MDP): A Python data processing framework. Frontiers in Neuro informatics. January (2008)

[12] S. Sonnenburg, G. R¨atsch, S. Henschel, C.Widmer, J. Behr, A. Zien, F. de Bona, A. Binder, C. Gehl, and V. Franc. The SHOGUN machine learning toolbox. Journal of Machine Learning Research. 11:1799–1802 (2010)

[13] I Guyon, S. R. Gunn, A. Ben-Hur, and G. Dror. Result analysis of the NIPS 2003 feature selection challenge. Proceedings of the 17th International Conference on Neural

Information Processing Systems. 2004 December; Vancouver, Canada: MIT press. (2004)

[14] Dua D, Graff C. UCI Machine Learning Repository. Available from: http://archive.ics.uci.edu/ml (2017)

[15] R. A. Fisher. The use of multiple measurements in taxonomic problems. Annals of Human Genetics. 7(2):179-188 (1936)

[16] R. O. Duda and P.E. Hart. Pattern Classification and Scene Analysis. John Wiley Sons: New York (1973)

[17] B. V. Dasarathy. Nosing around the neighbourhood: A new system structure and classification rule for recognition in partially exposed environments. IEEE Transactions on Pattern Analysis and Machine Intelligence. PAMI-2(1):67-71 (1980)

[18] G.W. Gates. The reduced nearest neighbor rule. IEEE Transactions on Information Theory. 18(3): 431–433 (1972)

[19] Tao, D.C., Tang, X.O., Li, X.L., Wu, X.D. Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence. 28(7), 1088–1099 (2006)

[20] Giorgio, V., Marco, M., Francesca, R. Cancer recognition with bagged ensembles of support vector machines. Neuro computing. 2004; 56: 461–466 (2004)

[21] Hyunsoo, K., Peg, H., Haesun, P. Dimension Reduction in Text Classification with Support Vector Machines. Journal of Machine Learning Research. 6:37–53 (2005)

[22] Bellotti, T., Crook, J. Support vector machines for credit scoring and discovery of significant features. Expert Systems with Applications. 36:3302–3308 (2009)

Hoijin Yoon received the B.S., M.S. and Ph.D. degrees in Computer Science from Ewha Womans University, Korea, in 1993, 1998 and 2004, respectively Dr. Yoon worked in Ewha as a full-time lecturer for 2 years after getting her Ph.D. In 2007, she joined the faculty of the Department of Computer Engineering at Hyupsung University, Hwaseung-si, Korea. She is currently an Associate Professor in the Department of Computer Engineering, Hyupsung University. She is interested in Software testing and Mutation Analysis.