Algorithmic Splitting: A Method for Dataset Preparation

KHALID M. KAHLOOT AND PETER EKLER
Department of Automation and Applied Informatics, Budapest University of Technology and Economics, 1117 Budapest, Hungary
Corresponding author: Khalid M. Kahloot (khalid.kahloot@aut.bme.hu)

This work was supported in part by NKFIH, Hungary, under the BME Artificial Intelligence TKP2020 IE Grant (BME IE-MI-SC TKP2020), in part by the Ministry of Innovation and the National Research, Development and Innovation Office within the framework of the Artificial Intelligence National Laboratory Programme, and in part by the National Research, Development and Innovation Fund of Hungary through the Centre for Higher Education and Industrial Cooperation—Research Infrastructure Development (FIEK 16) Funding Scheme under Project FIEK 16-1-2016-0007.

ABSTRACT The datasets that appear in publications are curated and have been split into training, testing and validation sub-datasets by domain experts. Consequently, machine learning models typically perform well on such split-by-hand prepared datasets. Whereas preparing real-world datasets into curated split training, testing and validation sub-dataset requires extensive effort. Usually, repetitive random splits are carried out and trained and evaluated on until reaching out a good score on the evaluation metrics. In this paper, an algorithmic method is proposed for preparing the sub-datasets splits for machine learning models. The objective of the proposed method is to achieve an evenly representative splits out of the dataset with standard and algorithmic way that reduce the perplexity of random splitting.

INDEX TERMS Data preprocessing, clustering algorithms, machine learning.

I. INTRODUCTION

Pattern recognition is considered as the main task of machine learning models (e.g. object features), therefore, the data which is fed to it determines the quality of its results. Generally, in order to train the model, this data is split into different subsets (sub-datasets), these sub-datasets are used for model training, testing and validation. In terms of model training, model parameters (e.g. node weights) are derived, and the learning process is controlled externally by the so called, hyper-parameters.

While the training sub-dataset is used to fit the model parameters, the validation takes a role of tuning the model hyper-parameter and to give an estimate of the skill level of the model, withathal testing dataset is used for estimating the final tuned model skill level [1]. However, estimating the model performance on the same training data will lead to biased estimates, thus both, validation and testing datasets, should be held out from the original dataset.

Commonly, these subsets are chosen and divided either split-by-hand or randomly [2], where the earliest requires extensive effort, the latter risks the lose of good feature representation. In real-world datasets, it is very exhaustive prepare training, validation, and testing sub-datasets which comprise a good representation of the features and categories within the dataset. Interestingly, machine learning models perform quite excellent on split by hand curated datasets, which is not the case in real-world data. The most common practice among data scientists is to split the dataset randomly with repetitive iterations until approaching a rational performance of their machine models and counting on the Law of Large Number (LLN) to be fulfilled.

Furthermore, in some cases, splits for training, validation, and testing could have biased features and categories, for instance, outlier crowds could be found in a split, which, probably, affects the model performance. Therefore, maintaining the representation of the original dataset in the sub-datasets with the less effort is desirable, this can be reached by replacing the random split with an administrable method. In this work, we try to split the data into the three previously mentioned sub-datasets by following mathematical approach, then to train the model and compare the results with ones retrieved from following the random splitting way. Moreover, the proposed technique is applied on different datasets with various complexities, then they are fed into multiple models in order to assess the performance.

The rest of the paper is organized as follows, next section is related work II, where similar purpose publications...
are presented. The main difference that they are ad-hoc and heavily depend on statistics. Section III demonstrates the algorithmic splitting methods in details. Section IV is showing the examined datasets IV-A, the dimension reduction results and labeling by clustering preview IV-B, how algorithmic splitting is assigning data-points to sub-datasets IV-C, training deep learning models IV-D, and experiments setup IV-E where the hyper-parameters are listed. Section IV-B conducts a comparison between two clustering methods and their effect on results. A conclusion is drawn in Section VI. Additional mathematical background is covered in Appendix A-VI and the metrics are explained in VI-B. The deep learning models that are used in this paper are briefly explained in Appendix B-VI-B. Finally, a visual illustration about the two-dimensional data reduction is shown in Appendix C-1 and some picked-up examples from the ranges are shown in Appendix C-2.

II. RELATED WORK

While there is no clear metthan repetitive randomization, few publications have talk about the effect of unbalanced splits. For instance, James et al. [2] explained the meaning of each split of data. The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set. The evaluation on validation set error rate is a proxy to get some sense of model performance on unseen real-world data.

In [3], Racz et al. have explored the effect of dataset size and train and test split ratios on the performance of classification models. Several combinations of dataset sizes and split ratios have been evaluated with five different machine learning algorithms. To study the effect, 25 performance parameters were calculated for each model then factorial ANOVA to compare the results. The machine learning models that have been evaluated are XGBoost, Naive Bayes, support vector machine (SVM), multi-layer feed-forward Neural Network, and probabilistic neural network (PNN). A visual way was more appealing beside numerical evaluation that showed a significant effect on multiple machine learning models XGBoost provided the best overall performance, while the standard Naive Bayes classifier was outperformed and more sensitive to differences in the size of the input dataset.

In [4], Racz et al. provided guidance on interpreting the results of Quantitative Structure–activity Relationship (QSAR) models. The challenge they addressed was to figure out the correct split of training and testing for the dataset that improves the ranking of performance parameters. Coefficient of concordance and correlation coefficient were used to get the distributions of the training and testing sets similar enough to each other.

In [5], Nguyen et al. have evaluated and compared the performance of different machine learning models such as Artificial Neural Network (ANN), Extreme Learning Machine (ELM), and Boosting Trees (Boosted) models considering the influence of various training to testing ratios in predicting the soil shear strength. Although the researchers are environmental engineers, they have done a great job in their investigation by dividing the datasets into the training and testing datasets for the performance assessment of models. The metrics that has been used are Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Correlation Coefficient (R). The predictive capability has been evaluated for each model under different training and testing ratios. Besides, Monte Carlo simulation was simultaneously carried out to evaluate the performance of the proposed models, taking into account the random sampling effect. Based on the statistical analysis, the best ratio was 70/30 for training and testing datasets. In addition, Monte Carlo simulations showed that the performance of the models is different under the random sampling effect over 1000 simulations.

III. PROPOSED METHOD: ALGORITHMIC SPLITTING

The ideal case scenario is when the data is perfectly aligning in a standard Gaussian distribution, which makes that tasks of splitting into training, validation, and testing sub-datasets an easy task by only sample out of each range of that distribution. The ranges have given names to indicate the vicinity to the mean and the mean plus two standard deviation. The lingering question is the percentage of sampling out of each range that will be landed in either training, validation, and testing sub-datasets. Three hyper-parameters are introduced for specifying the sample percentage out of each range as illustrated in FIGURE 1 and actually demonstrated in FIGURE 2.

![FIGURE 1. An illustration of split ranges on a Standard Gaussian Distribution, The $\mu$ and $\sigma$ are the mean and standard deviation of this distribution.](image1)

![FIGURE 2. MNIST dataset density distribution that is calculated on 3-dimensional data after reduction. The vertical lines are the separation of the three dimensions.](image2)
The three hyper-parameters are presented to control the splitting mechanism:
- \( \alpha \): percentage of sampling into training sub-dataset.
- \( \beta \): percentage of sampling into testing sub-dataset.
- \( \gamma \): percentage of sampling into validation sub-dataset.

In this paper, an algorithmic splitting method is proposed for evenly representing the dataset in training and validation sub-datasets. The ideal case scenario is to be able to estimate the distribution of the entire dataset, which requires high computational resources, especially with high-dimensional data. Instead, a transformation into lower dimensional data is made in order estimate the distribution of the data with reasonable amount of computational resources. By observing the shape of the graph of kernel density distribution (KDD), it appears clearly that the data is a mixture of Gaussian distributions. Empirically, the best fine-tuned shape of KDD is a mixture of standard Gaussian distributions, as shown in FIGURE 3. By transform the data into three-dimensional data, the KDD is tend to align into a mixture of three Gaussian distributions.

This method ensures that the sub-datasets have resembling data distributions to the original dataset. To achieve such goal, four steps have to be taken:
- First, a dimension reduction step is performed because it is almost impossible to inference from high-dimensional data. This step must use a deterministic transformation that processes the same results every time for a certain dataset.
- Second, a clustering step has to be applied on the assign data-points to clusters. The clustering labels are used heuristically through the splitting process. Therefore, a dimension reduction step is essential of the algorithmic method, which is followed by labeling by clustering prior to further processing.
- Third, splitting the clustered dataset into three separate ranges (median range, quartile range, and extreme range) for each cluster and gathering each range solely is done. These ranges keep the data original distribution maintained over three groups, thus, building a subset by picking data from these groups separately then combining them will reduce the randomness, and will ensure that the resulted subset will contain data distribution closer to the original data distribution. Hence, the model will be trained on subsets which are more representative of the original data, which will result in more stable and better performance. This is deployed in the next step.
- Four, preparing data which will be used for model training, hence, training and validation sub-datasets. In this step, random sampling from each range into training and validation sub-datasets is done, then concatenating the resulted sub-datasets from the three ranges. The sampling is performed by three hyper-parameters to determine the size of each sample.

### IV. EXPERIMENTS

The proposed algorithm is experimented on several datasets and applying different clustering methods. Furthermore, several models are trained and tested on the data to evaluate the performance with respect to the ground truth results. The examined datasets, dimension reduction technique, clustering methods, trained models, and the designed experiments are explained in more depth in this section.\(^1\)

#### A. EXAMINED DATASETS

The first step of this work is to choose the datasets which are examined, taking in consideration the variety of complexity between them, in order to increase the confidence degree in the results, and to avoid false conclusions. For this, five datasets are used:

1. Code can be found here: github.com/kkahloots/algorithmic splitting.
different datasets are used, which vary in complexity, in terms of dimension, size, and labels.

The examined datasets are MNIST [6] and Fashion-MNIST [7], both of them consist of total 70000 28 × 28 pixel images with 10 categories, handwritten digits and fashion articles respectively. In addition to CIFAR-10 [8] which contains 60000 colored 32 × 32 pixel images of 10 objects categories. Also, Small NORB [9] is used, it encloses 48600 images for 50 toys, each image is 96 × 96 pixel, the dataset has many labels, however, the object label is used which has five categories. Lastly, 60000 images from Shapes3d [10] dataset are used with their shape label (consists of four categories), the image size is 32 × 32 pixel. Each dataset loading results in two splits (training and testing), these two splits are either used or concatenated then randomly split into training and testing sub-datasets after shuffling. However, in case there is no pre-split (no original split) for the dataset, as in Shapes3d, the loaded dataset is shuffled and split into training and testing sub-datasets.

B. DIMENSION REDUCTION AND LABELING BY CLUSTERING

This step consists of two sub-steps, the first is the dimension reduction into three-dimension for data preparation prior to the clustering. For dimension reduction, Umap [11] is used because it is deterministic algorithm. The Umap is not only can produce the same results each run, but also work efficiently with few data-points [11]. In all our experiments, a random sample of only 10000 data-points are used, for each datasets, for the dimension reduction step. For labeling, two clustering algorithms are chosen to be applied on the three-dimensional data. Two types of clustering are examined: Spatial and Distribution-based clustering. Namely, Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [12] and Gaussian Mixture Models (GMM) [13] are applied for the labeling by clustering. The number desired clusters are dynamically chosen by HDBSCAN based on density. On the other hand, the number of clusters are set manually to match the original class labels in the original datasets. It doesn’t necessary mean any advantage to match or mismatch the original class labels unless it lead to a higher score in evaluation and that is something to be experimented on until figuring out the best number of clusters. For instance, the original class label and clustering by HDBSCAN are identical on MNIST dataset, as shown in FIGURE 3). On contrary, the number of class labels are different for other datasets, as shown in FIGURE 8.

C. ASSIGNING DATA-POINTS TO SUB-DATASETS

To decide the confidence intervals for assigning the data-points to ranges, at first, the dataset was split based on the mean and standard deviation of a normally distributed data-points, the results were not satisfying, since obviously the data is not normally distributed. To overcome this issue, Chebyshev’s inequality bound [14] was used (see Appendix A), to calculate the data-points certain distance from the mean, nonetheless, similar issue faced us, since the data-points are not centered at mean and the deviation is relatively large, in addition to the fact that Chebyshev’s inequality [14] is weaker than the bounds which can be obtained for the normal distribution.

To overcome any bias in the sub-datasets, a percentage-wise random sampling from each range is performed and distributed across the sub-dataset. Sampling is used to specify the cut points and to split the regions among the dataset in to three main ranges; median range, quartile range, and extreme range, by the parameters $\alpha$, $\beta$, $\gamma$ respectively. The capability of dealing with diverse distributions, which is the case in this work, due to its simplicity is a main advantage of using sampling over the other methods. Despite the fact that this simplicity, relatively, compromises the accuracy (see FIGURE 4).

D. TRAINING AND TESTING ON DEEP LEARNING MODELS

To guarantee relatively standard experiments, pre-trained models are used. The following three models are trained and tested, VGG [15], ResNet [16] and Inception [17].
FIGURE 4. A Preview of samples out of each range for MNIST and Small NORB datasets. Data-point from median range resembles clear handwriting digits for MNIST dataset. While for Small NORB dataset, the soldier’s figure is straight and his hands are totally visual. Data-points from median range are relatively easy to recognize by any machine learning model. On Contrary, data-points from extreme range are ill-shaped handwriting digits for MNIST dataset and unrecognizable because of the pose. The data-points from quartile range are somewhere in the middle between median range and extreme range data-points.

These architectures show great performance in the field of image data classification, and reached breakthroughs in terms of performance and efficiency, thus, they are chosen to assess the proposed algorithm performance. In each experiment, the pre-trained model is topped by dense layer with RELU activation. The last layer is another dense prediction layer with softmax activation. Hence, the pre-trained models parameters are not trained. Categorical cross entropy is used as a loss Adam is used as optimizer.

E. EXPERIMENT SETUP

Random sampling without replacement from each region is kept as training sub-dataset and validation sub-datasets, respectively. Then, the training chunks are concatenated together to form training sub-dataset, the same is done for the validation chunks which results in validation sub-dataset. Lastly, the model is trained and validated on these sub-datasets, and the performance is measured on the hold-out testing sub-dataset.

To cover all possible cases, twenty five different experiments are carried out. To establish a baseline, the ground truth splits are trained and tested on, which suppose to represent the best way of splitting the data. All experiments on the ground truth are repeated 1000 times and averaged. On the other hand, the datasets are split in completely random manner. All experiments on the random splits are repeated 1000 times and the worst are chosen. For the algorithmic splitting, the testing sub-dataset is held out before hand, then the splitting is applied on the other proportion, then the data from the median range is considered as training sub-dataset, and the concatenated data from quartile and extreme ranges is considered as validation sub-dataset. This scenario mimic the case when the model is trained on clean data (from median range), but it is tested on mixed data (hold-out sub-dataset). Table 1 shows the hyper-parameters of the algorithmic splitting method and a numbering of corresponding experiments to be able to refer to it.

TABLE 1. Hyper-parameters combination for algorithmic splitting.

| Experiment No. | α    | β    | γ    |
|----------------|------|------|------|
| 1              | 50%  | 10%  | 40%  |
| 2              | 45%  | 10%  | 45%  |
| 3              | 40%  | 10%  | 50%  |
| 4              | 35%  | 10%  | 55%  |
| 5              | 55%  | 10%  | 35%  |
| 6              | 50%  | 15%  | 15%  |
| 7              | 45%  | 15%  | 40%  |
| 8              | 40%  | 15%  | 45%  |
| 9              | 35%  | 15%  | 50%  |
| 10             | 45%  | 20%  | 35%  |
| 11             | 40%  | 20%  | 40%  |
| 12             | 35%  | 20%  | 45%  |
| 13             | 40%  | 25%  | 35%  |
| 14             | 35%  | 25%  | 40%  |
| 15             | 35%  | 30%  | 35%  |
| 16             | 20%  | 70%  | 10%  |
| 17             | 20%  | 10%  | 70%  |
| 18             | 70%  | 10%  | 20%  |
| 19             | 70%  | 20%  | 10%  |

Other experiments are carried out on extreme case in particular. A study of biased splitting is mimic by assigning all data-points of the extreme range to either training, testing or validation sub-datasets. Table 2 shows the percentages of sampling out of the three ranges with bias towards one the ranges each experiments. With respect to $\alpha$, $\beta$, and $\gamma$, the two-dimensional data is split into three ranges and the sampled randomly without replacement by a these three hyper-parameters.

TABLE 2. Bias sampling percentages of the hyper-parameters.

| Experiment No. | $\alpha$ | $\beta$ | $\gamma$ |
|----------------|----------|----------|----------|
| 20             | 27%      | 63%      | 10%      |
| 21             | 10%      | 63%      | 27%      |
| 22             | 63%      | 10%      | 27%      |
| 23             | 63%      | 27%      | 10%      |
| 24             | 10%      | 27%      | 63%      |
| 25             | 27%      | 10%      | 63%      |

V. RESULTS AND DISCUSSION

A. COMPARING HDBSCAN AND GMM

After performing all the aforementioned experiments on the five different datasets, interesting findings are noticed, and plenty of future work possibilities appear. The best performing hyper-parameters were presenting in experiments 4, 7, and 12 from Table 1. The overwhelming winning values of hyper-parameters are 10% out of median range, 35% out quartile range, and 55% out of extreme range. Experiments on random splitting have shown a poor performance compared to ground truth results.
TABLE 3. Accuracy results for algorithmic splitting with HDBSCAN.

|                  | Ground Truth | Random Split | Algorithmic Splitting |
|------------------|--------------|--------------|-----------------------|
|                  |              | 4*           | 7*                    | 12*                   |
| MNIST            | 79.0%        | 58.4%        | 80.6%                 | 80.4%                 | 79.2%                 |
| Fashion-MNIST    | 77.8%        | 33.2%        | 81.5%                 | 80.5%                 | 74.9%                 |
| CIFAR-10         | 76.8%        | 29.6%        | 72.3%                 | 72.2%                 | 71.5%                 |
| Small NORB       | 79.4%        | 27.3%        | 80.5%                 | 79.6%                 | 72.6%                 |
| Shapes3d         | 72.3%        | 32.6%        | 76.7%                 | 76.6%                 | 71.3%                 |

*Hyper-parameters of each experiment are detailed in Table 1.

TABLE 4. Accuracy results for algorithmic splitting with GMM.

|                  | Ground Truth | Random Split | Algorithmic Splitting |
|------------------|--------------|--------------|-----------------------|
|                  |              | 4*           | 7*                    | 12*                   |
| MNIST            | 79.0%        | 58.4%        | 75.1%                 | 75.6%                 | 75.7%                 |
| Fashion-MNIST    | 77.8%        | 33.2%        | 76.6%                 | 75.8%                 | 75.8%                 |
| CIFAR-10         | 76.8%        | 29.6%        | 55.9%                 | 55.8%                 | 53.9%                 |
| Small NORB       | 79.4%        | 27.3%        | 54.3%                 | 54.2%                 | 53.5%                 |
| Shapes3d         | 72.3%        | 32.6%        | 61.7%                 | 61.9%                 | 60.0%                 |

*Hyper-parameters of each experiment are detailed in Table 1.

FIGURE 5. Box plot for comparing the winning experiments. Algorithmic splitting with HDBSCAN shows higher variance and balanced distribution more than algorithmic splitting with GMM.

Table 4 shows the results of the same experiment setup but with GMM clustering algorithm. The ground truth splits outperformed all algorithmic splitting setups. Nonetheless, the algorithmic splitting method still performing better than random splitting. It is worth mentioning that the accuracy of the models are much worse with the Small NORB and Shape3d compared to MNIST, Fashion-MNIST and CIFAR-10 datasets. The winning values of hyper-parameters are also 10% out of median range, 35% out quartile range, and 55% out of extreme range. Algorithmic splitting method has shown higher accuracy for MNIST, Fashion-MNIST and CIFAR-10 datasets compared to the ground truth splitting. Moreover, algorithmic splitting method has shown very close accuracy for Small NORB and Shape3d compared to the ground truth splitting.

Comparing results from HDSCAN (Table 3) and GMM (Table 4), there is no significant difference between accuracy of both methods for simple datasets like MNIST and Fashion-MNIST. Nonetheless, both HDSCAN and GMM are still better than random splitting.

FIGURE 6. Accuracy of Small NORB training, testing and validation sub-datasets when choosing bias sampling percentages experiment 20 in Table 2.

FIGURE 7. Accuracy of testing sub-dataset when choosing bias sampling percentages experiment 20 in Table 2.

B. BIAS SAMPLING RESULTS

This section is referring to experiments that appears in Table 2. Most of the results are not worth mentioning except for experiment 20. Surprisingly, the accuracy score still performing relatively good in all datasets, as show in FIGURE 7. Clearly, HDBSCAN has scored higher accuracy compared with GMM. Although the variance of accuracy is more wider that HDBSCAN, the accuracy distribution of HDBSCAN is uniform and the mean is higher. The accuracy of training sub-dataset has scored an average of 69.6% and the testing sub-dataset has scored an average of 53.7% for the Small NORB dataset, as shown in FIGURE 6.

VI. CONCLUSION AND FUTURE WORK

Algorithmic splitting method has shown competitive results compared the ground truth splitting and outstanding results compared to random splitting. Algorithmic splitting with spatial clustering and distribution-based clustering are outperforming random splitting of the data. Algorithmic splitting with spatial clustering is outperforming the ground truth splitting, especially with complex datasets like Small NORB and Shapes3d. The spatial clustering has shown uniform distribution of accuracy scores and higher mean values more
than the distribution-based clustering in all experiments even for those experiments of bias sampling out the extreme and quartile ranges.

**APPENDIX A**

**A. MATHEMATICAL BACKGROUND**

Chebyshev’s inequality bound [14]: In probability theory, Chebyshev’s inequality bound, is considered as one of the most used inequality bounds, for a random variable with known variance. The inequality bounds are bigger than the bound used for a normally distributed variable, the standard deviation, but it gains its strength from the fact that it could be applied for any distribution, when the mean and the variance are known.

**B. METRICS**

- Precision: it mainly measures the degree of precision/accuracy of the model is terms of the values predicted positive, how many of them are actually positive.

\[
\text{Precision} = \frac{\text{Number of True Positive}}{\text{Total Number of Predicted Positive}}
\]

- Recall: it mainly measures the degree of accuracy of the model of predicting the actual positives with respect to the values which are really positive.

\[
\text{Recall} = \frac{\text{Number of True Positive}}{\text{Total Number of Actual Positive}}
\]
FIGURE 9. Showing examples from each range of the algorithmic splitting. Examples from median range are typical examples and easy to recognize. The examples from quantile range are rather recognizable but less definitive than the examples from the median range. Examples from extreme range are confusing and less well-formed than other ranges.

- F1-Score: It is used in order to seek balance between, previously mentioned metrics, Precision and Recall.

\[ F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \]

- Hamming loss: it is the fraction on the labels which are predicted incorrectly, and it is designed in case of multi class classification.

APPENDIX B

A. MODELS

- Very Deep Convolutional Networks for Large-Scale Image Recognition (VGG) [15] VGG model, mainly focus on an important aspect of ConvNet architecture, which is the depth of the network. The depth of the network supports up to nineteen layers (deep CNN). The accuracy of the network exceeded the state-of-art-accuracy in image recognition tasks. VGG is considered as one of the most used architectures in image recognition field and it reached high performance in 2014 ILSVRC [18].

- Inception Model [17] It is different network architecture than VGG [15], in which the depth and the width of its simple network allows the exceeding the performance of the conventional convolutional neural network. This comes with a highly computational cost. The Inception
model was introduced to reach a higher performance with lower cost, where the focus is to build a wider network instead of deeper, hence, it performs well with constrained computational abilities and memory. InceptionV3 was the first runner in 2015 ILSVRC [18].

- Residual Neural Network (ResNet) [16] ResNet is a deeper neural network. It was represented to gain the advantage of deep neural networks architecture while overcoming the drawbacks which face them in terms of training difficulty and complexity, by introducing the so called, learning residual functions in formulating the layers in the network. ResNet optimization complexity is lower than other deep neural networks, and the achieved accuracy is higher due to the increased depth. ResNet was the winner in 2015 ILSVRC [18].

APPENDIX C

A. TWO-DIMENSIONAL DATA PREVIEW
See Figure 8.

B. VISUAL EXAMPLES FROM THE RANGES
See Figure 9.

REFERENCES

[1] B. Ripley, *Pattern Recognition and Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 2007. [Online]. Available: https://books.google.hu/books?id=m12UR3QmLsqC

[2] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications In R*. New York, NY, USA: Springer-Verlag, 2013. [Online]. Available: https://www.springer.com/gp/book/9781461471370

[3] A. Rácz, D. Bajusz, and K. Héberger, “Effect of dataset size and train/test split ratios in QSAR/QSPR multiclass classification,” *Molecules*, vol. 26, no. 4, p. 1111, Feb. 2021. [Online]. Available: https://www.mdpi.com/1420-3049/26/4/1111

[4] A. Rácz, D. Bajusz, and K. Héberger, “Consistency of QSAR models: Correct split of training and test sets, ranking of models and performance parameters,” *SAR QSAR Environ. Res.*, vol. 26, nos. 7–9, pp. 683–700, Sep. 2015, doi: 10.1080/1062936X.2015.1084647.

[5] Q. H. Nguyen, H.-B. Ly, L. S. Ho, N. Al-Ansari, H. V. Le, V. Q. Tran, I. Prakash, and B. T. Pham, “Influence of data splitting on performance of machine learning models in prediction of shear strength of soil,” *Math. Problems Eng.*, vol. 2021, pp. 1–15, Feb. 2021.

[6] Y. LeCun and C. Cortes. (2010). *MNIST Handwritten Digit Database*. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[7] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms,” *CoRR*, vol. abs/1708.07747, pp. 1–6, Aug. 2017. [Online]. Available: http://arxiv.org/abs/1708.07747

[8] A. Krizhevsky, “Learning multiple layers of features from tiny images,” Tech. Rep., 2009.

[9] Y. LeCun, F. Jie Huang, and L. Bottou, “Learning methods for generic object recognition with invariance to pose and lighting,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jun. 2004, pp. 2–104.

[10] C. Burgess and H. Kim. (2018). *3D Shapes Dataset*. [Online]. Available: https://github.com/deepmind/3dshapes-dataset/

[11] L. McInnes, J. Healy, and J. Melville, “UMAP: Uniform manifold approximation and projection for dimension reduction,” Tech. Rep., 2020.

[12] L. McInnes, J. Healy, and S. Astels, “HDBSCAN: Hierarchical density based clustering,” *J. Open Source Softw.*, vol. 2, no. 11, p. 205, Mar. 2017, doi: 10.21105/joss.00205.

[13] X. He, D. Cai, Y. Shao, H. Bao, and J. Han, “Laplacian regularized Gaussian mixture model for data clustering,” *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 9, pp. 1406–1418, Sep. 2011.

[14] B. G. Amidan, T. A. Ferryman, and S. K. Cooley, “Data outlier detection using the Chebyshev theorem,” in *Proc. IEEE Aerosp. Conf.*, Mar. 2005, pp. 3814–3819.

[15] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” Tech. Rep., 2015.

[16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” *CoRR*, vol. abs/1512.00567, pp. 2818–2826, Jun. 2015. [Online]. Available: http://arxiv.org/abs/1512.00567

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet large scale visual recognition challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.