Dataset Augmentation for Machine Learning Tasks with Symbol Regression Program Synthesis

Ruchi Bhalani
Department of Computer Science
University of Texas at Austin
ruchi.bhalani@utexas.edu

***Abstract*—In this paper we propose a combination of Symbolic Regression and hill-climbing algorithm to generate artificial numerical and categorical training data for machine learning tasks. The generation of artificial data is useful for mitigating imbalanced datasets, protecting data that contains sensitive information, and keeping models from overfitting in the case of smaller datasets. We test our proposal on the Breast Cancer Wisconsin (Diagnostic) Data Set, the Credit Card Fraud Detection, and Pima Indians Diabetes Database benchmark datasets and show that Neural Net (NN), Deep Neural Net (DNN), Decision Tree (DT), and Support Vector Machine (SVM) classifiers reach approximately the same or higher accuracy, recall, and precision than models trained on the original data set.**

I. Introduction

Data augmentation, the artificial creation of training data for machine learning, is a widely studied field across machine learning disciplines [1]. While it is useful for increasing the generalization capabilities of a model by making it more robust to overfitting, it can also address many other challenges. Synthetic data generation can be helpful for oversampling minority classes as well as generating new data sets to preserve the privacy of sensitive original data (for example, medical records).

The first important motivation for synthetic data augmentation, oversampling the minority class, is relevant when training on data sets with severe class imbalances. It is common for databases to have underrepresented classes. For example, in credit card fraud databases, the ratio between normal and fraudulent transactions can be over 10K to 1, such as in the benchmark Credit Card Fraud Detection data set [2], which contains only 492 fraudulent transactions out of a data set with 284,807 total transactions. The same is possible when analyzing data sets with patient medical information and the number of healthy patients (or benign tumors) is much higher than the number of afflicted patients (or malignant tumors). Classification tasks struggle with imbalanced data sets since the program would demonstrate a low error even if it misclassified all the minority class data points. Augmentation both increases the representation of the minority class and can help avoid overfitting and benefit generalization to unseen data.

The second significant motivating use case for synthetic data augmentation is to avoid using the original data for privacy reasons since it is possible that a data set contains sensitive information, and working on it directly leaves it vulnerable to data misuse. For example, medical records may contain lots of personal information about patients, even without their names. One possible approach to this issue is to avoid using original data to train the model and rather generate a synthetic data set on which to train the model.

There are several machine learning-based techniques used for data augmentation, including scaling, adding noise to data samples, or using Generative Adversarial Networks (GANs) to generate artificial data [3]. Although there have been many ML-based models such as SMOTE and ADASYN [4] [5], one potential downfall of these systems is that they may make models more prone to overfitting, where the model becomes too specialized to the train set and doesn't generalize well to a held-out test set (previously unseen data) [6].

One novel potential approach to data set augmentation is by using a combination of machine learning techniques, genetic programming, and symbolic regression to synthesize new data entries that mimic the original data set.

Symbolic regression aims at searching the space of mathematical expressions that best fit a given dataset [7]. Symbolic regression models have been widely used in industrial empirical modeling due to their interpretable nature [8] [9], however, due to the high dimensional search space of mathematical expressions that can describe a specific dataset, symbolic regression is a complex combinatorial problem [10]. Therefore, traditional symbolic regression approaches have been based on genetic algorithms.

Genetic programming is a common method for performing symbolic regression that relies on the use of random constants to scale predictions [11]. With the exception of gaps at the boundaries causing erratic variance, genetic programming performs well when applied to benchmark problems, particularly with small training samples. Standardized genetic programming works well on difficult problems with little need for data augmentation.

Therefore, for problematic datasets (severe class imbalances, too small to generalize), we propose symbolic regression using genetic programming, a promising approach to synthesize programs that can generate new dataset entries that mimic the relationships and correlations in the original dataset. This is framed as a synthesis from holes in programs, in which the unknown variables that make up the expression that fits the data set are the holes, or the "unknowns", that need to be synthesized by looking through a search space.

II. Approach

A. Numerical Variable Generation

The data augmenter goes through the original data set and generates a user-specified number of examples that has the same statistical profile and distribution as the original data set.

The data in each column is scaled to the range [0,1]. Then a symbolic regressor is fit to the original data set to synthesize an expression that fits the distribution of the data set. For our purposes, we employ the `gplearn` Python API[1] Our symbolic regressor architecture has the following parameters:

1) Initialization: When starting a genetic programming run, the first generation is unaware that there is a fitness function that needs to be maximized. These programs are naive and usually a random mix of the available functions and variables in the search space, so they will generally perform poorly [11].

- **Population size = 5000.** This specifies the size of the population of individuals, or potential solutions, in each generation of the genetic algorithm.

2) Selection: Now that we have a population of programs, it is important to decide which of these programs will evolve into the next generation. This is done through tournaments.

- **Tournament size = 50.** This specifies the size of tournament selection, which involves selecting random individuals from the population and then choosing the best one as the parent for the next generation.

3) Evolution: The fitness measure is used to find the fittest individual, or "winner" of each tournament, but before they can graduate to the next generation, genetic operations are performed on them.

- **P Crossover = 0.7.** This specifies the probability of exchanging genetic material between two parents in the genetic algorithm.
- **P Subtree Mutation = 0.1.** This specifies the probability of a subtree mutation, which involves randomly selecting a subtree from an individual in the population and replacing it with a new, randomly generated subtree.
- **P Hoist Mutation = 0.1.** This specifies the probability of hoist mutation, which involves selecting a random subtree from the winner of a tournament. A random subtree of that subtree is selected and then "hoisted" to the original subtree's location in order to form the next generation's offspring.
- **P Point Mutation = 0.1.** This specifies the probability of a point mutation, which involves selecting a random node in an individual's tree and replacing it with a new, randomly generated node.

4) Termination: There are two possible ways that the evolution process will stop.

- **Generations = 50.** This specifies the number of generations the algorithm will run for.
- **Stopping criteria = 0.01.** This specifies a stopping criterion for the algorithm based on the mean absolute error of the best individual in the population.

5) Bloat: A common phenomenon in genetic programming algorithms is known as *"bloat"*, in which the program sizes grow larger and larger with no significant fitness improvement, leading to longer computational times with no solution benefit [12].

- **Parsimony coefficient = 0.001.** This controls the trade-off between model complexity and goodness of fit. A higher value means that simpler models are favored over more complicated ones. Because we are prioritizing the quality of the generated data rather than the interpretability of the synthesized expression, we select a low parsimony coefficient.
- **Maximum samples = 0.9.** This specifies the proportion of samples to use to train each generation. Increasing the amount of subsampling performed on data results in more diverse results for individual programs.

B. Categorical Variable Generation

Synthetic data set augmentation requires a search of a large number of possible solutions to find one that satisfies all constraints. This search space is a *combinatorial* search space. Hill-climbing search is a search algorithm used to go through large search spaces, such as that of the N-queens problem, and find a solution by allowing us to optimize a certain objective function [13] [14]. It has been found that local search methods for SAT outperform more traditional backtrack strategies and that many combinatorial problems can be encoded as Boolean SAT problems [13], so hill-climbing algorithms provide a good mechanism for solving combinatorial problems.

We use hill-climbing algorithm to generate synthetic categorical variables for the data set. For each set of variables, the algorithm perturbs the distribution of the categorical variable by randomly swapping two values in the distribution. Then, it computes the correlation of the perturbed categorical variable with the numerical variables. if the correlation improves with the perturbation, the algorithm updates the categorical variable with the new distribution and repeats this process for ten iterations.

III. Evaluation

First, we evaluate the quality of the data on its own through an examination of the augmented distribution and a comparison to the original data set distribution for each feature. Then, we evaluate the original and augmented data sets on three different case studies using three benchmark data sets and evaluate the results on four different models using 5-fold cross-validation.

A. Models

For each of the case studies, we evaluate the given data set using the following models: Decision Trees (DT), Support Vector Machines (SVM), Neural Networks (NN), and Deep Neural Networks (DNN). These models were chosen because

[1] https://github.com/trevorstephens/gplearn

they are known to be prone to overfitting. Synthetic data augmentation should reduce overfitting on the train set, and boost performance on the train set [15]. We evaluate each of the models on each of the data sets using an 80-20 split for test-train sets, and 5-fold cross-validation.

For decision trees, we used the default model parameters. The architectures of the rest of the models are detailed below.

1) SVM: The SVM uses a linear kernel function and assumes that the classes are linearly separable, rather than the default radial basis function kernel, which is more generalized [16].

2) NN: This is a sequential NN architecture that consists of three layers: two fully connected (Dense) layers and an output layer with a single neuron. The first hidden layer has 12 neurons with a ReLU activation function, the second hidden layer has 8 neurons with an activation function. The output layer has a single neuron with a sigmoid activation function, which maps the output to a range between 0 and 1, which is especially useful for binary classification problems. The model is compiled with the binary cross-entropy loss function and the Adam optimizer [17]. The model is trained with a batch size of 100 for 150 epochs.

3) DNN: This DNN architecture consists of five fully connected (Dense) layers and an output layer with a single neuron. The first, second, third, and fourth hidden layers contain 12, 16, 8, and 4 neurons respectively, all using the ReLU activation function. The output layer has a single neuron with the sigmoid activation function. Similar to the neural network architecture mentioned in the prior section, this model is compiled with the binary cross-entropy loss function and the Adam optimizer, and trained with a batch size of 100 over 150 epochs. This is a relatively deep network with five total hidden layers, and the number of neurons in each layer is reduced. Because of its architecture, this model might be able to learn more complex patterns in the data in comparison to our shallower network. Additionally, since the number of neurons is reduced, it may be less prone to overfitting [18].

B. Data Quality Evaluation

In order to evaluate the quality of the augmented data set specifically in regards to how well it mimics the distribution of the original data set, we use a measure of Kullback-Leibler divergence, which quantifies in bits how close a probability distribution $p = \{p_i\}$ is to a model or candidate distribution $q = \{q_i\}$ [19]:

$$D_{KL}(p||q) = \sum_i p_i \log_2 \frac{p_i}{q_i} \tag{1}$$

A common technical interpretation is that the KL divergence is the coding penalty associated with selecting a distribution q to approximate the true distribution p [20]. We calculate the average KL divergence across features for each data set and then take a closer look at the distributions for individual features.

C. Case Study A: Insufficient Size Data sets

One reason that data sets might require synthetic data augmentation is if there aren't enough samples in the original data set to train a model on. Augmenting the original data set could potentially allow the model to better learn the correlation between variables and predict the class labels better.

The experiment for this study will be conducted on the **Breast Cancer Wisconsin (Diagnostic) Data Set** [21]. The features of this data set are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe the characteristics of the cell nuclei present in the image. The data set has a target class to determine if the cancer is benign or malignant. There are 357 benign and 212 malignant samples, which is a relatively small amount of total examples for a model to learn a pattern.

In order to increase the size of the data set, we augment the train data set by 150% with synthetic examples and observe if this synthetic data is able to increase the quality of model predictions on the test set.

D. Case Study B: Class Imbalance

Another potential use case for synthetic data augmentation is in cases where the data set contains a severe class imbalance. This leads to classification algorithms mislabeling the minority classes because they don't have enough data to train on.

For these experiments, we will be using the **Credit Card Fraud Detection** data set [2], which contains transactions made by credit cards in September 2013 by European cardholders. It presents transactions that occurred in two days. Out of the 284,807 transactions, there are only 492 frauds. This data is highly imbalanced since the positive class (frauds) accounts for only 0.172% of all transactions.

In order to mitigate this class imbalance, we perform a version of oversampling from the minority class using synthetic data rather than naive resampling. In order to properly evaluate whether the minority classes are being classified correctly, our test set contains 50% frauds and 50% legitimate transactions. Then, we perform data augmentation on the positive class entries in the train set, augmenting the train set by 10 times the original number of minority samples in the train set. This way, we increase the presence of the minority class with synthetic data so that the model doesn't naively always predict the negative class.

E. Case Study C: Privacy Protection

The last use case we will be evaluating is synthetic data augmentation for privacy protection. In cases where the original data set contains potentially sensitive information and we want to use as little of it as possible to train a model, synthetic data would be helpful.

Most of these potentially-sensitive data sets would likely be medical information, so for these experiments, we use the **Pima Indians Diabetes Database** [22], which consists of 8 independent variables and 1 target dependent class that represents if the patient has diabetes or not. It is composed of 768 total samples, of which 268 patients present the disease.

To simulate training on potentially sensitive data, we test the performance of models on an entirely synthetic data set. Rather than augmenting the data set in this example, we fit the symbolic regressor to the original data and use it to synthesize entirely new data that makes up 100% of the new train set.

IV. Results

A. Data Quality Evaluation

In this project, we are using an augmented data set (q) to approximate a true data set (p). Using KL divergence as a method of similarity, we calculate the divergence between each feature of the data sets, for each benchmark data set we will be evaluating and we will take an average. These averages per data set can be seen in Table I. We also take a closer look at the distribution of each feature in the Pima Indians Diabetes Database [22] for the original and augmented data sets in Figure 1: Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, and Age. This qualitative comparison of histograms (the distributions of the original and augmented data sets seem to be visually similar) along with a relatively low average KL divergence, demonstrates that the synthetic data generated is able to mimic the original data set fairly well.

B. Case Study A: Insufficient Size Data sets

The performance metrics for all four models trained on the original Breast Cancer Wisconsin (Diagnostic) Data Set [21] and the augmented data set, which contains the original train data as well as generated synthetic data the size of 150% of the original train set, is visible in Table II.

The augmented data set shows an increase in accuracy and recall for 3 out of the 4 models: DT, NN, and DNN, as well as an increase in precision for SVM and NN. In many of the remaining metrics for the remaining models, the original and augmented data sets performed similarly. The only metric that seems to be worse for the augmented data set is the precision for DT and DNN. Further study is required to analyze why these specific models seem to be suffering a hit in precision when the remainder of the models perform better on the original data. Class labels are assigned to the samples using categorical variable generation since the labels are "M" for Malignant or "B" for benign, so further tuning of the climbing-hill algorithm may result in better precision performance.

This boost in accuracy and recall indicates that a greater amount of training data allowed the model to better learn the relationship between the features and the class label, and that the generated synthetic data was faithful enough to the original data that it improved overall performance on the held-out test set rather, as opposed to flooding the original data set with fuzzy data relationships that distort the model's perception on the data.

This is aligned with our hypothesis that a greater number of training samples would allow for better performance across the models.

C. Case Study B: Class Imbalance

The performance metrics for all four models trained on the original Credit Card Fraud Detection [2] data set and the augmented data set is visible in Table III.

This data set contained an incredibly severe class imbalance, with fraudulent transactions making up only 0.172% of the transactions. This led to incredibly poor precision and recall for all models on the original data set. Since precision measures the accuracy of positive predictions while recall measures the completeness of positive predictions, this indicates that the model was overfitting to the train set by never predicting the positive class. The class imbalance in this data set is preventing the models from learning the relationships between features for fraudulent transactions, which are much more important to correctly identify than legitimate transactions.

To mitigate this, we combined the technique of oversampling from the minority class and synthetic data generation, augmenting the original train set with 10 times the number of minority samples previously present. This provides the benefit of mitigating class imbalance without naive resampling. This resulted in a similar accuracy to the original data set (due to the class imbalance, the model would receive 0.828 accuracy even if it predicted the negative class label for all samples). However, the benefit of data augmentation is obvious in the results for precision and recall. Precision and recall metrics shot up from 0 (never predicting the minority class) to values in the range 0.79-0.89 for recall and 0.95-0.98 for precision. This enormous difference in performance demonstrates that generating data from the minority class allows the models to successfully learn the relationships between attributes for fraudulent transactions, allowing them to correctly which transactions are members of the minority class and correctly identify a large proportion of the total minority class samples.

This also follows from our hypothesis, which was that generating synthetic data to oversample the minority class would allow for more rich model learning and better performance in predicting the positive class.

D. Case Study C: Privacy Protection

The performance metrics for all four models trained on the original Pima Indians Diabetes Database [22] is visible in Table IV. Although this data set does not contain sensitive information, a common use case for private data sets likely involves personal medical information, so for the purposes of our experiments, we have decided to treat the information in the original data set as private. Therefore, we train on 100% synthetic data, using none of the data from the original data set in the "augmented" data set, so as to protect information privacy.

Using this approach, the models perform fairly similarly on both the original and "augmented" (completely synthetic) data sets. Accuracy is similar for both data sets, performing slightly better on the original data set for the NN model. For recall, the augmented data set performs a little better on DT and DNN, and the original data set performs better on SVM

	Breast Cancer Wisconsin [21]	Credit Card Fraud Detection [2]	Pima Indian Diabetes [22]
Average KL divergence across all features	0.0071	0.0059	0.0068

TABLE I: To calculate the similarity between the original data set feature distribution and the augmented data set feature distribution, we take the average KL divergence over all the features in the data set.

Model	Accuracy		Recall		Precision	
	Orig.	Augm.	Orig.	Augm.	Orig.	Augm.
DT	0.7142	0.7597	0.5964	0.7045	0.6181	0.5636
SVM	0.7532	0.7662	0.7179	0.7111	0.5090	0.5818
NN	0.7142	0.7532	0.6170	0.6666	0.5272	0.6181
DNN	0.6038	0.7012	0.4634	0.5918	0.6909	0.5272

TABLE II: Performance of all models on the Breast Cancer Wisconsin (Diagnostic) Data Set [21], with a 150% increase in train data for the augmented train set. Performance metrics were obtained by taking the mean from 5-fold cross-validation. If the original data set performs better on that metric for that model, the cell is highlighted in red. If the augmented/synthetic data set performs better on that metric for that model, the cell is highlighted in green. If they perform about the same (within 0.01 of each other), then neither cell is highlighted.

Model	Accuracy		Recall		Precision	
	Orig.	Augm.	Orig.	Augm.	Orig.	Augm.
DT	0.9544	0.9536	0	0.8546	0	0.9535
SVM	0.9743	0.9876	0	0.7957	0	0.9754
NN	0.9986	0.9983	0	0.8867	0	0.9846
DNN	0.9786	0.9857	0	0.8975	0	0.9746

TABLE III: Performance of all models on the Credit Card Fraud Detection [2], with an oversampling from the minority class of 10x the samples present in the original train set. Performance metrics were obtained by taking the mean from 5-fold cross-validation. If the original data set performs better on that metric for that model, the cell is highlighted in red. If the augmented/synthetic data set performs better on that metric for that model, the cell is highlighted in green. If they perform about the same (within 0.01 of each other), then neither cell is highlighted.

and NN. For precision, the original performs better on DT and the augmented performs better on SVM and NN. This slight variation is attributed to randomness and error, since the differences are not very significant, and seem to be fairly random.

Overall, the synthetic data set seems to approximate the original data set fairly well, even maintaining variation in recall and precision. This indicates its ability to mimic the original data set while not copying it exactly, which is why the metrics seem to vary across the board, even though they are overall very similar. This aligns with our hypothesis that our synthetic data would be able to protect sensitive information by learning patterns and synthesizing new data with symbolic regression, which allows machine learning models to be trained on this new data with similar model performance and minimal risk for data abuse or exploitation.

Model	Accuracy		Recall		Precision	
	Orig.	Augm.	Orig.	Augm.	Orig.	Augm.
DT	0.9427	0.9342	0.8435	0.8643	0.9234	0.9012
SVM	0.9385	0.9412	0.8064	0.7096	0.9615	0.9823
NN	0.9834	0.9394	0.8234	0.7542	0.9562	0.9724
DNN	0.9734	0.9634	0.8357	0.8853	0.9762	0.9723

TABLE IV: Performance of all models on the Pima Indians Diabetes Database [22], with synthetic data making up 100% of the "augmented" train set. Performance metrics were obtained by taking the mean from 5-fold cross-validation. If the original data set performs better on that metric for that model, the cell is highlighted in red. If the augmented/synthetic data set performs better on that metric for that model, the cell is highlighted in green. If they perform about the same (within 0.01 of each other), then neither cell is highlighted.

V. Conclusion

Data set augmentation is a widely studied topic with varying approaches that span fields of study. It is especially helpful in cases where machine learning data sets do not contain a sufficient number of samples for models to learn a pattern from, instances of class imbalance so severe that the models are unable to pick up on attributes of the minority class and identify them in the test set, or cases where data sets contain private, personal, or otherwise sensitive information that must be protected.

Although there currently exist machine learning approaches to augmenting data sets, many of these existing approaches run the risk of generating data that exacerbates the risk of models overfitting on the train sets and generalizing poorly to held-out (previously unseen) test sets. Symbol regression program synthesis, which synthesizes an expression by searching the space of mathematical expressions that best fit a given data set, is the approach that we develop in this project to generate synthetic data from an existing data set. Genetic programming is the method we employ for performing symbolic regression since it performs well even on smaller data sets.

Additionally, for features that do not fit in a mathematical expression, such as categorical variables, we employ hill-climbing algorithm, which has been shown to perform well on combinatorial problems. For each set of variables, the algorithm perturbs the distribution of the categorical variable and then computes its correlation with the numerical variables, keeping the perturbation if it resulted in a higher correlation.

With these techniques for synthetic data augmentation, we test its performance on three use cases: a data set of insufficient size, a data set with a severe class imbalance, and a data set with potentially private information.

For Case Study A, augmenting the data set with additional

samples resulted in an increase in performance across all the models. For Case Study B, oversampling from the minority class with synthetic data generation resulted in a jump in recall and precision performance across models, since they were no longer overfitting and naively only predicting the negative class. For Case Study C, training on a 100% synthetic test set resulted in a very similar performance to training on the original data set, which means that it was successful in mimicking the patterns seen in the original data set and could be used in instances to approximate potentially sensitive or classified data.

In this project, we presented a novel method for data set augmentation that circumvents the need for naive approaches and overcomes the limitations of approaches that may fall prey to overfitting.

Future work may include testing a larger variety of benchmark data sets with even more extreme class imbalances, or purely categorical variables, in order to evaluate how performance differs. Additionally, we could run further experiments to evaluate how well the privacy of sensitive data sets is preserved in synthetic data generation.

REFERENCES

[1] M. Bayer, M.-A. Kaufhold, and C. Reuter, "A survey on data augmentation for text classification," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–39, 2022.

[2] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection: A realistic modeling and a novel learning strategy," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3784–3797, 2018.

[3] "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, no. 1, p. 321–357, jun 2002.

[5] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008, pp. 1322–1328.

[6] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," 2017.

[7] I. A. Abdellaoui and S. Mehrkanoon, "Symbolic regression for scientific discovery: an application to wind speed forecasting," *CoRR*, vol. abs/2102.10570, 2021. [Online]. Available: https://arxiv.org/abs/2102.10570

[8] M. Kotanchek, G. Smits, and E. K. Vladislavleva, *Trustable symbolic regression models: Using ensembles, interval arithmetic and pareto fronts to develop robust and trust-aware models*, 01 1970, pp. 201–220.

[9] S. Sun, R. Ouyang, B. Zhang, and T.-Y. Zhang, "Data-driven discovery of formulas by symbolic regression," *MRS Bulletin*, vol. 44, pp. 559–564, 07 2019.

[10] N. Benabbou, C. Leroy, and T. Lust, "An interactive regret-based genetic algorithm for solving multi-objective combinatorial optimization problems," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 03, pp. 2335–2342, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/5612

[11] C. A. Owen, G. Dick, and P. A. Whigham, "Standardization and data augmentation in genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 6, pp. 1596–1608, 2022.

[12] T. H. Chu, Q. U. Nguyen, and V. L. Cao, "Semantics based substituting technique for reducing code bloat in genetic programming," in *Proceedings of the 9th International Symposium on Information and Communication Technology*, ser. SoICT '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 77–83. [Online]. Available: https://doi.org/10.1145/3287921.3287948

[13] B. Selman and C. P. Gomes, *Hill-climbing Search*. John Wiley Sons, Ltd, 2006. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/0470018860.s00015

[14] L. Hernando, A. Mendiburu, and J. A. Lozano, "Hill-climbing algorithm: Let's go for a walk before finding the optimum," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, 2018, pp. 1–7.

[15] R. Roelofs, V. Shankar, B. Recht, S. Fridovich-Keil, M. Hardt, J. Miller, and L. Schmidt, "A meta-analysis of overfitting in machine learning," in *Neural Information Processing Systems*, 2019.

[16] B. Schölkopf, K. K. Sung, C. J. C. Burges, F. Girosi, P. Niyogi, T. A. Poggio, and V. N. Vapnik, "Comparing support vector machines with gaussian kernels to radial basis function classifiers," *IEEE Trans. Signal Process.*, vol. 45, pp. 2758–2765, 1997.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[18] D. A. Winkler and T. C. Le, "Performance of deep and shallow neural networks, the universal approximation theorem, activity cliffs, and qsar," *Molecular informatics*, vol. 36, no. 1-2, p. 1600118, 2017.

[19] J. Shlens, "Notes on kullback-leibler divergence and likelihood," *CoRR*, vol. abs/1404.2000, 2014. [Online]. Available: http://arxiv.org/abs/1404.2000

[20] T. M. Cover and J. A. Thomas, *Elements of Information Theory, 2nd edition*, 2006.

[21] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[22] J. Smith, J. Everhart, W. Dickson, W. Knowler, and R. Johannes, "Using the adap learning algorithm to forcast the onset of diabetes mellitus," *Proceedings of Annual Symposium on Computer Applications in Medical Care*, vol. 10, pp. 261–265, 11 1988. [Online]. Available: https://www.kaggle.com/uciml/pima-indians-diabetes-database

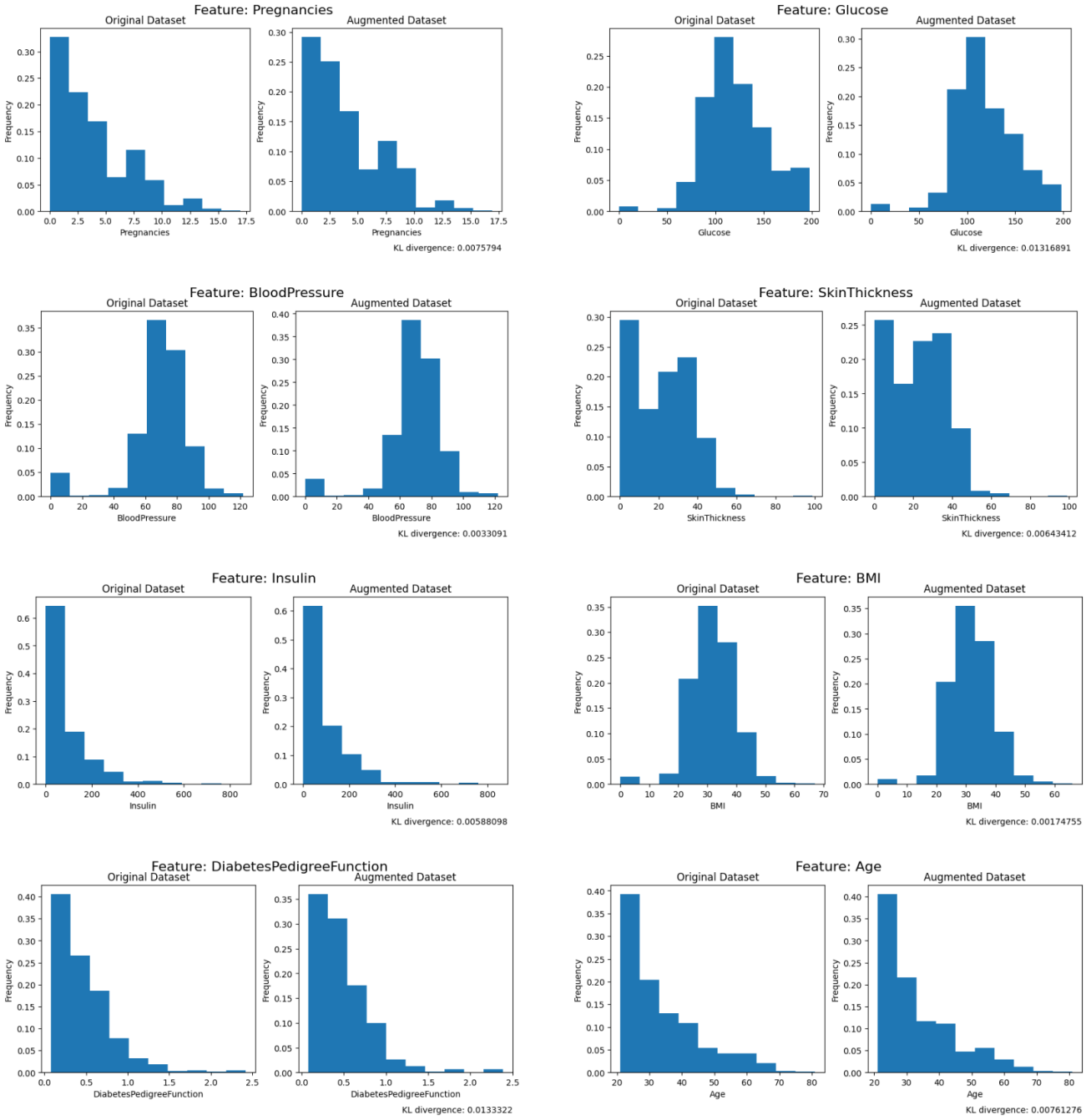

Fig. 1: Distributions for the original and augmented versions of the Pima Indians Diabetes Database for nine features. KL divergence is also shown. This demonstrates that the augmented data set is able to serve as an approximation of the original data set with very little "coding penalty" [20].