Using a meta-model to compensate for training-evaluation mismatches

Dylan Lamprecht[1][0000−0002−2172−3991] and Etienne Barnard[2][0000−0003−2202−2369]

Multilingual Speech Technologies (MuST), North-West University, South Africa; and CAIR, South Africa.

Abstract. One of the fundamental assumptions of machine learning is that learnt models are applied to data that is identically distributed to the training data. This assumption is often not realistic: for example, data collected from a single source at different times may not be distributed identically, due to sampling bias or changes in the environment. We propose a new architecture called a meta-model which predicts performance for unseen models. This approach is applicable when several 'proxy' datasets are available to train a model to be deployed on a 'target' test set; the architecture is used to identify which regression algorithms should be used as well as which datasets are most useful to train for a given target dataset. Finally, we demonstrate the strengths and weaknesses of the proposed meta-model by making use of artificially generated datasets using a variation of the Friedman method 3 used to generate artificial regression datasets, and discuss real-world applications of our approach.

Keywords: Generalization · Meta-model· Mismatched distributions· Robustness· Machine learning· Tree-based models.

1 Background

The past decade has witnessed great successes in machine learning, in various areas [17], including image classification [3], speech recognition [13] and recently natural language processing [4].

This model can then be deployed, and its performance can be measured on a *test* or *evaluation set* - the essential characteristic of Machine Learning is that the training and evaluation sets are distinct, and the goal is to achieve good *generalization* on the evaluation set.

In classical learning theory, one assumes that the training and evaluation data are drawn from the same probability distribution [15]. However, that assumption is often not realistic for practical applications. For the sake of convenience, training data for an image recognition system may be collected in a particular location, whereas the system may be deployed (and evaluated) more widely. A more troubling example was recently reported for a system that predicts the probability that people are likely to re-offend in the USA criminal justice system: the algorithms show an implicit bias when predicting for two different ethnic groups, because it had been trained with biased data [16]. The current assumption that is required for the analysis of machine learning models is that models are built such that the dataset the model trains on and the set which the model is tested against have the same statistical distribution [11]. This research wishes to identify statistically similar, but not identical distributions, and show that good machine learning performance can be achieved under those conditions.

Our goal is to demonstrate the meta-model's ability to identify the presence of a mismatch between datasets as well as identify which datasets have the least change in distribution. This scenario is important in several applications of pattern recognition, such as speech recognition where clean training data must be used to recognize noisy speech, or computer vision when simulated images are used as training for real-world recognition.

We are interested in problems where this mismatch is unavoidably significant, namely in applications where several relevant 'proxy' training sets are available, and none of these is guaranteed to match the 'target' evaluation set. Such tasks are common in certain sectors of the financial industry, where several prepackaged databases may be available and the goal is to use one or more of those for prediction on a new dataset. Since our simulated data are based on a regression task from that industry, we only show results for a particular family of regression models. However, it will become clear that the same approach can be applied to more general tasks.

2 Related work

Generalization is the ability for machine learning algorithms to perform well on a set of unseen samples. This ability can be enhanced by giving additional information to sets of training data. The method frequently used to improve invariance to change is to make use of the transformation of a dataset [7]. The rationale behind a linear transformation is that as long as a model can adapt to a linear transformation without a drop in performance, the model can adapt to realistic changes and hence can generalize better. Specifically, one focuses on determining how to compensate for a covariate shift. However, this does not allow for a shift in the target variable or the relationships between the features and the target variables.

In our context, it is not clear how one can ensure that the generated data approximates the distribution of the evaluation dataset; thus, this approach will generally bias the model to predict on the generated data.

To compensate for this, Generalized robust risk minimization (GRRM) creates an ensemble of training sets in order to have an increase in generalization performance. GMMR is a variation of Robust risk minimization (RRM), which is a robust variation of Empirical risk minimization (ERM) [11]. This model makes use of aggregating ensembles of training data. Since ERM drastically degrades when noise is introduced [11], GRRM is a method to improve the generalization performance of models as demonstrated in its ability to adapt to noise, covariate shifts and weak multi-labels [8].

The most frequent method for measuring differences in distributions prior to running them is by making use of Distance Metric Learning. Distance Metric Learning (DML) is an important method to use as a baseline approach in many machine learning problems. This quantifies the difference in distributions between the two datasets ('training' and 'evaluation' or 'proxy' and 'target'). This makes the algorithms adaptable to changes and identifies which datasets are distributed more closely to one another [2]. Various methods of utilizing distance metrics have been proposed – as an example, we discuss Consistent distance metric learning (CDML) [2].

To compensate for variation between datasets, CDML [2] makes use of distance metrics to balance the cost function of a dataset by making use of feature importance pairs, using importance sampling methods from supervised learning for metric learning. Two important weighting strategies were tested:

– Estimating the importance weights of data instances before calculating importance weights for instance pairs.
– Estimating the importance of instance pairs directly.

This method has been shown to be effective for classification tasks on clustering problems for real-world and artificial datasets [2].

We have reviewed several methods that have been proposed to deal with mismatches between training and test data. Although methods of making models more robust are useful when dealing with mismatches between datasets, they rely on each dataset being similar to the target set, while our proposed method focuses on measuring which datasets are the closest related to a target dataset.

3 Methodology

We now describe our approach to the mismatch problem, which employs a metal-model. The purpose of the meta-model is to use selected features from trained models to predict the performance of the model on an unseen dataset. This is particularly useful since it provides insight into which datasets and regression algorithms will perform well, without a validation set, as is required in the case when the datasets are distributed significantly differently.

3.1 Algorithm overview

The algorithm trains a set of models using various combinations of proxy data, target data and regression algorithms as outlined in Figure 1. For each trained model, we measure various 'meta-features' and then create a meta-model which predicts the accuracy of a particular proxy-target-algorithm combination based on those meta-features. This meta-model can then be used to predict which proxy and regression algorithm should be used for a given target set.

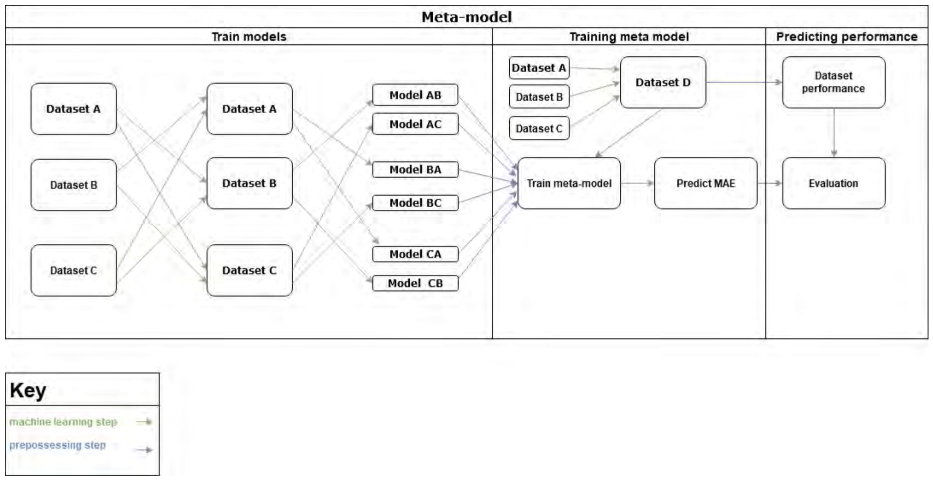

Fig. 1. Flow diagram of process for meta-model training and evaluation. During the training models phase, we collect the meta-data used to train the meta-model by forming each possible test-train pair of datasets and train models for each test-train combination. We then proceed to the process summarized in 'Training meta model', where we process all the features (outlined in Subsection 3.3) of the models trained in the 'train models' phase as meta-data. We use the meta-data as features to train the meta-model to predict the mean absolute error (MAE) for a given model. As a test set, we train each model using each other dataset and applied it to the features of an unseen test dataset to create the 'meta-features'. We then compare the actual MAE to the predicted MAE of the unseen test set to evaluate the performance of the meta-model.

3.2 Training and using the meta-model

Training the models To collect relevant training data for the meta-model (meta-data), we trained a variety of models and extracted the relevant information (as described in Subsection 3.3) on the results of the training and test sets. For each dataset, each other dataset was used as a training set to form a test-train pair. There were six regression algorithms used to train six models for each dataset combination specified in Subsection 3.4 'Input to the model'.

Training the meta-model To train the meta-model we made use of the meta-data of models and tested the results against the target dataset. The process of training the meta-model consisted of using the meta-data as features (as described in Subsection 3.3) and using the mean absolute error (MAE) as the target feature. We then trained a model using the catboost regression algorithm to predict the MAE of a model tested on an unseen dataset. To test the performance, we used a dataset not used in the training set and predicted performance of each dataset and each model on this unseen dataset.

3.3 Meta-model features

To create a meta-model, we measured a variety of metrics of the trained models, related to predicted ranges, distance metrics and general information about each of the features and then predicted the MAE for each model by using the following classes of features:

- 'Meta-features' about the features used for training the underlying models: these include the bins (outlined in the next paragraph) of each of the features in the training and test sets, their means and standard deviations, and several distance metrics between the pair of datasets (Kullback–Leibler divergence [12], Kolmogorov–Smirnov test [5], Hellinger distance [6] and Wasserstein distance [14])
- 'Meta-features' about the predicted target of the underlying models: bins (outlined in the next paragraph) of the predicted values for both sets, their means and standard deviations, and the sum of the predicted values.

Bins We created twelve bins, which is done by sorting values of a feature or predicted target in ascending, dividing them into equal parts and then add a value representative of the interval each value falls under. Then for each feature and predicted target we had an equal number of entries for all the features for both training and test set and used the boundary of each bin as features. Hence, these features are the relevant percentiles of the features for the two sets.

3.4 Regression algorithms

Various tree-based methods (Catboost, Extra tree regressor, Gradient boosted regressor, Light GBM, Random forest and XGBoost) were employed as regression algorithms. These tree-based regressors were chosen and parameters were chosen from literature [10] along with state-of-the-art variations of gradient boosted trees [1]). Based on informal experiments, we believe that these are reasonable parameter choices for each of the algorithms. Since our aim is to estimate the ranking of these algorithms, rather than deriving a single best algorithm, it is not necessary to obtain the very best setting for any of the parameter sets. It is also evident that the performance of the algorithms is relatively similar as demonstrated in table 4

4 Experiments

We investigate the performance of the proposed meta-model approach by showing the results obtained on a group of artificial datasets. We generated related datasets by using the mathematical function

$$tan^{-1}\left(\frac{x_2x_3 - \frac{1}{x_2x_4}}{x_1}\right) \tag{1}$$

as proposed by Friedman [1] (his method '3'). This method was chosen since it was the method that generated datasets most closely resembling the real-world data (which we cannot share due to confidentiality constraints) this method was originally developed for. For the related datasets we wish to demonstrate how the meta-model adapts to the following situations:

– Generating noise: We generated a uniform distribution between 0 and 1 ($U(0,1)$), multiplied the result with the noise factor and then added it to step three in Subsection 4.1 of the Friedman 3 implementation. For each dataset we added the amount of noise equal to the numeric order of the dataset to each dataset. Here we test how robust the meta-model is to minor changes in data.

$$tan^{-1}\left(\frac{x_2x_3 - \frac{1}{x_2x_4}}{x_1}\right) \times N(0,1) \times c \times dataset\ number \tag{2}$$

– Scale changes: We scaled each of the features generated in step 1 to create several unique datasets. For example, if the scale change was five and there are five datasets, the datasets would have the following ranges for the features [(0,0.2), (0.2,0.4), (0.4,0.6), (0.6,0.8), (0.8,1)]. Here we are primarily testing for whether the meta-model can identify which datasets are relatively close to each target dataset in terms of scale changes.

[1] The code is available on Github, under the project name meta-model.

– Data shifts: We multiplied the dataset number with data shift factor and added the result to step three in the Friedman 3 implementation.

$$tan^{-1}\left(\frac{x_2x_3 - \frac{1}{x_2x_4}}{x_1}\right) + c \times dataset\ number \tag{3}$$

– A combination of scale changes and data shifts: We combined the dataset from both scale changes and data shifts, in order to study how robust the meta-model was to combined changes.
– Model selections: We ran the model with different underlying regression algorithms. This was done as a method to determine which algorithms performed better across datasets with a scale change of 2.
– Dataset size: We tested how well the meta-model performs when the underlying datasets are trained on different numbers of underlying elements, to test how many samples are required in a dataset for the meta-model to be able to estimate performance well.

This will probe the meta-model's ability to generalize, by demonstrating the effects that different data alterations have on the meta-model and how the meta-model adapts to change. This will be used as a proxy for how real-world datasets are differently distributed and how the meta-model adapts to the changes.

4.1 Friedman 3 implementation

We generated 10,000 elements for each dataset, each element consisting of four features x_1, x_2, x_3 and x_4 as well as a target. To generate the features we implemented the following steps:

1. For each feature, we uniformly generated a value between 0 and 1. For each dataset, we generated the features with the seed used to initialize the random-number generator set equal to the number of the dataset (e.g. dataset 0 generates on seed 0, dataset 1 generates on seed 1 etc.)
2. Then we applied the following transformations to each feature
   – $x_1 = feature\ 1 \times 100$
   – $x_2 = feature\ 2 \times 520\pi + 40\pi$
   – $x_3 = feature\ 3$
   – $x_4 = feature\ 4 \times 10 + 1$
3. We finally applied the following transformation to get the target for each element:

$$target = tan^{-1}\left(\frac{x_2x_3 - \frac{1}{x_2x_4}}{x_1}\right) \tag{4}$$

4.2 Metrics measured

To evaluate the performance of the trained meta-model, we made use of the following metrics.

- Regression performance: We compared the true MAE to the predicted MAE using the correlation coefficient. The reason we are using correlation coefficient instead of a standard MAE or R^2 is since the modifications change the target values, which would indicate a larger error in terms of value in many cases, but because the emphasis of this paper is to find similarity, the relative order is more important and the correlation coefficient is able to measure the relative value without being influenced as much by the changes in for example generative noise.
- Ranking datasets performance: By using the MAE, we set up two lists with the datasets ranked from the best performance to the worst performance, using the actual and the predicted MAE. Then we used the Kendall Tau test [9] to determine relative similarity (on a scale from -1 to 1) of the two rankings.

4.3 Results

We now report on the results that were obtained when datasets generated according to these protocols were processed with the algorithm described in Section 3. For each setting of the generation parameters, we measure the correlation coefficient between the actual and predicted MAEs for each dataset-regressor combination using Leave-one-out cross validation on each of the 11 datasets. We also evaluate the accuracy of the meta-model in predicting the relative rankings of the different dataset combinations, using the Kendall Tau test [9].

	10	20	50	100
Metric	Correlation coefficient	Correlation coefficient	Correlation coefficient	Correlation coefficient
Average	0.9827	0.9712	0.9529	0.9454
Standard deviation	0.0321	0.0466	0.0881	0.1002
Median	0.9981	0.9988	0.9983	0.9983
Metric	Kendall Tau	Kendall Tau	Kendall Tau	Kendall Tau
Average	0.9515	0.9232	0.8828	0.8949
Standard deviation	0.0285	0.0797	0.1164	0.1094
Median	0.9667	0.9556	0.9111	0.9111

Table 1. Scale changes

	10	100	1000	10000
Metric	Correlation coefficient	Correlation coefficient	Correlation coefficient	Correlation coefficient
Average	0.8363	0.9261	0.9817	0.9857
Standard deviation	0.2848	0.1678	0.0217	0.0130
Median	0.9879	0.9923	0.9934	0.9907
Metric	Kendall Tau	Kendall Tau	Kendall Tau	Kendall Tau
Average	0.6929	0.8101	0.9515	0.9475
Standard deviation	0.1233	0.1688	0.0464	0.0556
Median	0.7333	0.8222	0.9556	0.9556

Table 2. Dataset sizes

	0.01	0.02	0.05	0.1
Metric	Correlation coefficient	Correlation coefficient	Correlation coefficient	Correlation coefficient
Average	0.4478	0.5043	0.5987	0.6082
Standard deviation	0.6096	0.5610	0.4201	0.3957
Median	0.7372	0.7081	0.8010	0.7560
Metric	Kendall Tau	Kendall Tau	Kendall Tau	Kendall Tau
Average	0.3778	0.3818	0.4869	0.4626
Standard deviation	0.5440	0.5123	0.3120	0.3428
Median	0.4667	0.5556	0.6000	0.6000

Table 3. Data shift

	Catboost	Light GBM	XGBoost	GBT	Random forest	Extra tree	all regression algorithms
Metric	Correlation coefficient	Correlation coefficient	Correlation coefficient	Correlation coefficient	Correlation coefficient	Correlation coefficient	Correlation coefficient
Average	0.9863	0.9886	0.9903	0.9594	0.9874	0.9860	0.9748
Standard deviation	0.0146	0.0119	0.0067	0.0449	0.0119	0.0132	0.0202
Median	0.9930	0.9934	0.9912	0.9833	0.9938	0.9880	0.9805
Metric	Kendall Tau	Kendall Tau	Kendall Tau	Kendall Tau	Kendall Tau	Kendall Tau	Kendall Tau
Average	0.8909	0.8990	0.9071	0.8343	0.9394	0.9030	0.8454
Standard deviation	0.0608	0.0565	0.0464	0.1072	0.0850	0.0623	0.0542
Median	0.8667	0.8667	0.9111	0.8222	0.9556	0.9111	0.8441

Table 4. Regression algorithm selection

		10	20	50
Dataset section	Metric	Correlation coefficient	Correlation coefficient	Correlation coefficient
Scale changes	Average	0.9785	0.9752	0.9695
	Standard deviation	0.0191	0.0203	0.0395
	Median	0.9862	0.9829	0.9831
		Kendall Tau	Kendall Tau	Kendall Tau
Scale changes	Average	0.8459	0.8286	0.8277
	Standard deviation	0.0385	0.0664	0.0962
	Median	0.8381	0.8476	0.8476

Table 5. Combining data shifts and scale changes to measure the effects on scale changes

		10	20	50
Dataset section	Metric	Correlation coefficient	Correlation coefficient	Correlation coefficient
Data shifts	Average	0.7416	0.7054	0.7621
	Standard deviation	0.2924	0.4539	0.3241
	Median	0.8873	0.9094	0.8869
		Kendall Tau	Kendall Tau	Kendall Tau
Data shifts	Average	0.5749	0.5697	0.5905
	Standard deviation	0.0621	0.1166	0.0853
	Median	0.5810	0.6381	0.6000

Table 6. Combining data shifts and scale changes to measure the effects on data shifts

	0	0.01	0.02	0.05	0.1
Metric	Correlation coefficient	Correlation coefficient	Correlation coefficient	Correlation coefficient	Correlation coefficient
Average	0.9853	0.9855	0.9798	0.9616	0.9663
Standard deviation	0.0143	0.0152	0.0218	0.0454	0.0357
Median	0.9926	0.9921	0.9912	0.9819	0.9845
Metric	Kendall Tau	Kendall Tau	Kendall Tau	Kendall Tau	Kendall Tau
Average	0.9273	0.9232	0.9192	0.8949	0.8505
Standard deviation	0.0637	0.0491	0.0623	0.0802	0.0999
Median	0.9111	0.9111	0.9111	0.9111	0.8667

Table 7. Additive noise

Processing	Elements	10 datasets	20 datasets	30 datasets	50 datasets
Single core processing	100	51.19	195.98	436.93	1,211.22
	1,000	92.88	367.83	788.92	2,152.32
	10,000	721.62	2,759.05	5,550.93	15,369.38
Multicore processing	100	8.98	28.65	63.04	174.33
	1,000	12.54	48.22	102.56	279.13
	10,000	86.44	316.97	690.90	1,898.71

Table 8. Computation

5 Discussion

The following findings were obtained from the experiments discussed in Section 4 outlined in tables 1- 8.

The meta-model is able to adapt to differences in scale changes, and larger-scale changes result in less accurate meta-model predictions. This is to be expected since the model is unable to reproduce the relationship completely; the greater the difference in the scale, the more significant the inability to reproduce the relationship is as shown in Table 1.

From Table 2 it is evident that the meta-model's performance is the inverse of the scale changes in Table 1 i.e. greater numbers of samples in the datasets result in better predictions of the MAE. The meta-model improves logarithmically with a greater number of elements in a dataset: for this task, we need at least 1,000 samples to achieve a correlation coefficient of better than 0.95.

By implementing data shifts and breaking the relationship between the features and targets in Table 3, the prediction performance of the meta-model is reduced significantly. As the data shift increases, the model is able to predict *relative* MAEs more accurately: although in absolute terms the MAE becomes worse as the data shift increases, the correlation coefficient increases since the meta-model responds more accurately to larger changes.

Table 4 shows the effect when the meta-model is trained on models with different underlying regression algorithms. We specifically looked at the regression algorithms [Catboost, Light GBM, XGBoost, gradient boosted tree, random forest, Extra tree regressor] with hyperparameters as mentioned in Subsection 3.4. In general, the majority of regression algorithms perform similarly, the only two noticeable exceptions are gradient boosted trees, which has a lower correlation coefficient as well as a lower Kendall Tau ranking, and the random forest algorithm is able to rank the models (using Kendall Tau) 0.03 better than models trained with any other regression algorithm.

Table 5 We combined scale changes with data shifts (a standard 0.01 data shift) to test how well the meta-model adapts to different types of changes in different datasets. For scale changes combining the results with the data shifts did hurt the Kendall Tau ranking by around 0.1 for each correlation. However, the correlation coefficient was less sensitive to scale changes in comparison to the results obtained for only using scale changes in Table 1. Since the dataset

now combines both scale changes and data shifts, the overall effect is somewhere intermediate to the effects of each of the separate modifications.

Table 6 For the data shifts when combining scale changes with data shifts (a standard 0.01 data shift), the meta-models did perform better in comparison to Table 3, where the correlation coefficient improved by around 0.3 and the Kendall Tau ranking improved by 0.2. This is probably due to the dataset containing the scale changes, which are datasets the meta-model could predict more successfully.

When comparing the effect of additive noise on a meta-model in Table 7, we again found the expected behaviour: since the noise breaks the correlation between features and targets, more accurate prediction is achieved at lower noise levels.

We demonstrate the computational requirements of the meta-model, by documenting the architecture's performance in Table 8, we used a Ryzen 2600X CPU (containing 6 CPU cores with 12 threads) and the time was measured in seconds. In general, the computational performance followed two trends, the first is that the process benefits greatly with the number of cores and that the computational performance scales with the number of cores. The second is that the computational performance between dataset sizes can be described using the formula below

$$Model_2 \approx Model_1 \left(\frac{Datasets_2}{Datasets_1} \right)^2 \tag{5}$$

Where:

$$\begin{aligned} Model &= \text{The computation required to run the meta-model} \\ Datasets &= \text{The number of dataset used to train the meta-model} \end{aligned}$$

6 Conclusion

The meta-model has proven to be valuable in both providing an ordered ranking of the datasets and predicting how accurate each model would be for a given test dataset. For scale changes, we compared different levels of scale changes and the greater the level of scale change the worse the model performs. The meta-model can perform well with underlying models that are trained on very few elements; however, the performance of the meta-model improves with more elements in the underlying datasets. Creating a data shift does hurt model performance significantly; however, the level of change did not significantly affect the meta-model's performance. The meta-model's performance did not differ significantly when trained with different underlying regression algorithms, the only notable exception being gradient boosted trees. When applying scale changes with data shifts, the additional models harm the performance of the scale changes by a greater degree than the maximum scale change, and the data shifts perform better by a significant amount. Adding generated noise does harm the model's ability to predict, although it is able to compensate for the change for relatively small amounts of noise.

This method was originally adapted from a real world application on real world data (the nature of which can not be disclosed because of commercial confidentiality), and the datasets generated reflect the size and number of features in the real-world data. An interesting direction for future work would be to investigate combinations of datasets for training: we currently only predict the performance for each individual 'proxy' set, but optimal performance may well require combining several sets. Computing a meta-model prediction for each combination of sets is computationally prohibitive, so a more insightful approach is required.

In terms of deploying the system in practice, the architecture provides a summary of the predicted MAE's for each model and dataset for a given test set. From this, a user can select from a few models and use the predicted MAE as a proxy for the model's accuracy. This allows a user to set up a range of probable values for a given dataset along with a method to quantify the probability of each outcome.

References

1. Breiman, L.: Bagging predictors. Machine learning **24**(2), 123–140 (1996)
2. Cao, B., Ni, X., Sun, J.T., Wang, G., Yang, Q.: Distance metric learning under covariate shift. In: Twenty-Second International Joint Conference on Artificial Intelligence (2011)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
4. Jordan, M.I., Mitchell, T.M.: Machine learning: Trends, perspectives, and prospects. Science **349**(6245), 255–260 (2015)
5. Lilliefors, H.W.: On the Kolmogorov-Smirnov test for normality with mean and variance unknown. Journal of the American statistical Association **62**(318), 399–402 (1967)
6. Lindsay, B.G.: Efficiency Versus Robustness: The Case for Minimum Hellinger Distance and Related Methods. Annals of Statistics **22**(2), 1081–1114 (06 1994)
7. Ling, J., Jones, R., Templeton, J.: Machine learning strategies for systems with invariance properties. Journal of Computational Physics **318** (05 2016). https://doi.org/10.1016/j.jcp.2016.05.003
8. Mazuelas, S., Perez, A.: General Supervision via Probabilistic Transformations. arXiv e-prints arXiv:1901.08552 (Jan 2019)
9. Noether, G.E.: Why Kendall Tau? Teaching Statistics **3**(2), 41–43 (1981)
10. Olson, R.S., La Cava, W., Mustahsan, Z., Varik, A., Moore, J.H.: Data-driven Advice for Applying Machine Learning to Bioinformatics Problems. arXiv e-prints arXiv:1708.05070 (Aug 2017)
11. Osama, M., Zachariah, D., Stoica, P.: Robust Risk Minimization for Statistical Learning. arXiv e-prints arXiv:1910.01544 (Oct 2019)
12. Perez-Cruz, F.: Kullback-Leibler divergence estimation of continuous distributions. In: 2008 IEEE International Symposium on Information Theory. pp. 1666–1670 (2008)
13. Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., Vesely,

K.: The Kaldi Speech Recognition Toolkit. In: IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. IEEE Signal Processing Society (Dec 2011), iEEE Catalog No. CFP11SRW-USB
14. Vallender, S.S.: Calculation of the Wasserstein Distance Between Probability Distributions on the Line. Theory of Probability & Its Applications **18**(4), 784–786 (1974)
15. White, H.: Artificial neural networks: approximation and learning theory. Blackwell Publishers, Inc. (1992)
16. Yapo, A., Weiss, J.: Ethical Implications of Bias in Machine Learning (01 2018). https://doi.org/10.24251/HICSS.2018.668
17. Yu, A.W.: Effective and Efficient Learning at Scale. Ph.D. thesis, Carnegie Mellon University (2019)