Explaining Low Dimensional Representation, a reproduction

Anonymous Author(s)
Affiliation
Address
email

Reproducibility Summary

Scope of Reproducibility

This report covers our reproduction of the paper 'Explaining Low dimensional Representation' [7] by Plumb et al. In this paper, a method (Transitive Global Translations, TGT) is proposed for explaining different clusters in low dimensional representations of high dimensional data. They show their method outperforms the difference between the means (DBM) method, is consistent in explaining differences with few features and matches real patterns in data. We verify these claims by reproducing their experiments and testing their method on new data. We also investigate the use of more complex transformations to explain differences between clusters.

Methodology

We reproduce the original experiments using their source code. We also replicate their findings by re-implementing the authors' method in PyTorch [6]and evaluating on two of the dataset used in the paper and two new ones. Furthermore, we compare TGT with our own extension of TGT, which uses a larger class of transformations.

Results

We were able to reproduce their results using their code, yielding mostly similar results. TGT generally outperforms DBM, especially when explanations use few features. TGT is consistent in terms of the features to which it attributes cluster differences, across different sparsity levels. TGT matches real patterns in data. When extending the types of functions used for explanations, performance did not improve significantly, suggesting translations make for adequate explanations. However, the scaling extension shows promising performance on the modified synthetic data to recover the original signal.

What was easy

The easiest part was running the existing code with the pre-trained model files. The original authors had set up their code base in an organized manner with clear instructions.

What was difficult

The first difficulty that we encounter was finding the right environment. The source code depends on deprecated functionality. The clustering method they used, had to be re-implemented for us to use it in our replication. Another difficulty was the selection of clusters. The authors did not prove a consistent method for selecting clusters in a latent space representation. When retraining the provided models, we get a latent space representation different to the original experiments. The clusters have to be manually selected. The metrics that they used to evaluate their explanations are also depend on the clustering. This means that there is some variability in the exact verification of reproducibility.

Communication with original authors

We asked the original authors for clarification on how to choose the ϵ hyper-parameter. However, it became apparent that we had misread, and the procedure is indeed adequately reported in the paper.

Submitted to ML Reproducibility Challenge 2020. Do not distribute.

1 Introduction

The *curse of dimensionality* [3] is a long-standing problem in Machine Learning. Data in many domains and applications (e.g. Bioinformatics) has high-dimensional representations. Finding patterns in such high-dimensional data is a challenging task. To this end, *dimensionality reduction* [11] techniques have greatly helped in data-analysis, information extraction, building computational models, and in doing inference. Given an input $X \in \mathbb{R}^d$, dimensionality reduction learns a function $r : X \rightarrow x, \;\; x \in \mathbb{R}^m$, where $m << d$. Such a dimensionality reduction function r naturally arises in deep learning due to the expressivity and representational power of neural networks. The goal of r is to encode useful knowledge about the input space, thus providing distinctive information in the transformed output x. This results in "clusters" or "groups of points" in the transformation space. The downside of this exercise, however, is that the output space is usually non-interpretable. There is usually no easy way to know what information is present in the transformed points x_s and what sort of distinctive knowledge they contain.

In this work, we reproduce the paper 'Explaining Groups of Points in Low-Dimensional Representations' by Plumb et al. [7]. This paper proposes a method for explaining different clusters in latent space representation. They look at the problem of explaining the points in the latent space representation through the lens of Interpretability in Machine Learning. We reproduce their findings and expand upon them with our own extension. We extend their research by applying their method to a larger class of explanation functions and testing their method on new datasets. We further investigate the efficacy of the explanations using a probing classifier [2].

2 Methodology

Counterfactual Explanations [10] have emerged as an active research area in the field of Interpretable Machine Learning. A counterfactual explanation is defined as the smallest perturbation to the input that would change the output of a machine learning model. As such, these explanations are promising as they can provide suggestive recourse to the beneficiary in a machine learning based decision system. As an interpretable machine learning problem, Plumb et al. [7] aim to find such counterfactual explanations in order to explain the differences between the groups in latent space. To this end, they employ the function r itself to find what perturbation δ needs to be made in the input $x \in \mathbb{R}^d$ so that $r(x + \delta)$ belongs to the different target group. The goal is to find the *global* explanations that apply to the whole group as opposed to the *local* explanations which explain only individual examples [4]. Furthermore, the explanations need to be *sparse* for them to be interpretable by practitioners. Finally, these explanations should be be both *symmetric* and *transitive*. To obtain these explanations, the authors propose the algorithm called, Transitive Global Translations (TGT), explained hereafter.

Following the previous notation, let $r : \mathbb{R}^d \mapsto \mathbb{R}^m$ denote our dimensionality reduction function, where d is the dimensionality of the input space and m is the latent space's dimensionality. Suppose $X_i, X_j \subset \mathbb{R}^d$ get mapped to the clusters $R_i, R_j \subset \mathbb{R}^m$ respectively. The goal is to define the transformation $t_{i \mapsto j} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ on $x \in X_i$ as $x^{'} = t_{i \rightarrow j}(x)$, so that $r(x^{'}) \in R_j$, or equivalently $x^{'} \in X_j$.

The proposed algorithm TGT considers the transformations of the form $t_{i \rightarrow j}(x) = x + \delta_{i \rightarrow j}$. To find the optimal parameters of the transformation function, authors imply a compressed-sensing based objective function as below:

$$l(\delta_{i \rightarrow j}) = \|r(t_{i \rightarrow j}(\bar{X}_i)) - \bar{R}_j\|_2^2 + \lambda \|\delta_{i \rightarrow j}\|_1 \tag{1}$$

where $\lambda \|\delta_{i \rightarrow j}\|_1$ is a regularization term to incentivize sparser explanations, and $\bar{X}_i \in \mathbb{R}^d$ and $\bar{R}_j \in \mathbb{R}^m$ denote the means of the clusters in the input space and latent space respectively. Given clusters $0, 1, \ldots, n$, we get a total of $\frac{1}{2}n(n+1)$ transformations.

Furthermore, the transformations $t_{i \rightarrow j}$ has to adhere to several mathematical properties. Namely, for any clusters i, j, k these transformations should be : a) Symmetric, i.e. $t_{i \rightarrow j} = {t_{j \rightarrow i}}^{-1}$ and b) Transitive, i.e. $t_{j \rightarrow k} \circ t_{i \rightarrow j} = t_{i \rightarrow k}$. From these properties it follows that $t_{i \rightarrow i}$ is the identity function $\mathcal{I}$ as

$$t_{i \rightarrow i} = t_{i \rightarrow 0} \circ t_{0 \rightarrow i} = t_{i \rightarrow 0} \circ {t_{i \rightarrow 0}}^{-1} = \mathcal{I} \tag{2}$$

We define this condition as *self-similarity*. Furthermore, the group of translations is uniquely defined by $t_{0 \rightarrow 1}, \ldots, t_{0 \rightarrow n}$, because for any i, j:

$$t_{i \rightarrow j} = t_{0 \rightarrow j} \circ t_{i \rightarrow 0} = t_{0 \rightarrow j} \circ {t_{0 \rightarrow i}}^{-1} \tag{3}$$

Plumb et al. [7] compare their method against the naive baseline of Difference Between the Means (DBM). With DBM, each transformation is still a translation: $t_{i \rightarrow j}(x) = x + \delta_{i \rightarrow j}$. However, now $\delta_{i \rightarrow j} = (\bar{X}_j - \bar{X}_i)$. We also use this as a baseline for comparison in this report.

Since translations are a very narrow class of functions, we expanded upon the research by investigating other transformations that still satisfy the GCE requirements. We investigate the transformations of the form $t_{0\to i}(x) = \exp(\gamma_{0\to i}) \odot x + \delta_{0\to i}$. These always have a well defined inverse, given by ${t_{0\to i}}^{-1}(x) = \exp(-\gamma_{0\to i}) \odot (x - \delta_{0\to i})$ and only have $\mathcal{O}(d)$ parameters. The inclusion of scaling could enhance performance, while the necessary components of GCE are maintained.

2.1 Metrics to evaluate Global Counterfactual Explanations

To measure the efficacy of the transformation function $t_{i\to j}$, the authors propose two metrics, coverage and correctness. The **Coverage** ($cv(t_{i\to j})$) is the fraction of points $a \in R_j$ for which there is a point $b \in X_i$ such that $\|r(t_{i\to j}(b)) - a\|_2 < \epsilon$, i.e.

$$cv(t_{i\to j}) = \frac{1}{|R_j|} \sum_{a \in R_j} \mathbb{I}\left[\exists b \in X_i | \|r(t_{i\to j}(b) - a\|_2 < \epsilon\right] \tag{4}$$

The **Correctness** ($cr(t_{i\to j})$) is the fraction of points $b \in X_i$ for which there is some $a \in R_j$ such that $\|r(t_{i\to j}(b)) - a\|_2 < \epsilon$, i.e.

$$cr(t_{i\to j}) = \frac{1}{|X_i|} \sum_{a \in R_j} \mathbb{I}\left[\exists a \in R_j | \|r(t_{i\to j}(b) - a\|_2 < \epsilon\right] \tag{5}$$

Note that both these metrics have the hyperparameter ϵ which is to be chosen carefully. When $i = j$ we do not count the point itself, there must be some other point within distance ϵ. [1]

Furthermore, the **Similarity** metric measures the consistency of the explanations at different sparsity levels. Given two explanations e_1, e_2 where e_1 is more sparse than e_2, the similarity of e_1 and e_2 is defined as

$$sim(e_1, e_2) = \frac{\sum_i |e_1[i]| \mathbb{1}(e_2[i] \neq 0)}{\|e_1\|_1} \tag{6}$$

This is equal to 1 if e_1 uses a subset of the features that e_2 uses. By definition, DBM always has similarity 1.

3 Scope of Reproducibility

We investigate the following claims from the original paper:

1. In terms of the average correctness and coverage, TGT performs equally well or better than the DBM method. This remains true, especially for sparser explanations.
2. TGT explanations have similarity close to 1. It is consistent in which features it uses for explanations across different sparsities.
3. TGT correctly identifies known causal structure in data.
4. Furthermore, TGT explanations are consistent. When altering the dataset by adding a copy of a cluster with a specific feature altered, TGT recovers the modification with little change to the other explanations.

4 Methodology of Reproducibility

We make use of the code made available by the original authors [2] for our pilot investigative study. We first verify that the provided models and explanations stay true to the claims made in the paper. We further retrain their models on the provided dataset. We also made our own PyTorch [6] implementation to perform our extension experiments.

4.1 Model description

We identify that the scope of the original paper is to explain clusters in the low-dimensional representations. However, obtaining meaningful and discernible low-dimensional representations is an active area of research. The original authors employ a t-SNE [9] objective based Variational Autoencoder (called, henceforth, as scVIS) [5] as the r function. They make use of library[3] by the original scVIS authors in their implementation. We also implement this model in Pytorch

[1] We use this definition to set the value for epsilon, as explained in the *Methodology* section of the original Paper.

[2] https://github.com/GDPlumb/ELDR

[3] `https://github.com/shahcompbio/scvis`

for our experiments. However, we deliberately decide not to match the model implementation exactly. This is done to study the model-agnosticism of the TGT algorithm. By design, TGT should be able to explain the clusters for any differentiable r function. However, we maintain that r should give discernible latent representations with preserved global structure in the data. In our implementation of the scVIS library, we therefore do not employ the hyperparameters and the training settings from the original library.

4.2 Dataset Description

We reproduced the findings of the authors on the four datasets that they used. We used two of these datasets as well as two new ones to test our PyTorch implementation. All datasets can be found on our github repository.

1. **Single cell RNA** [8]: This dataset has 13166 features. We use the same number of clusters at the original authors, 18 in this case.
2. **UCI Boston housing** This dataset has 506 entries with 13 features. We use 6 clusters for both reproduction and replication.
3. **UCI Heart disease** This dataset has 303 entries with 13 features and 1 binary label. We used 8 clusters in the reproduction and 4 in the replication. The data was normalized to be in the range [0, 1].
4. **UCI Iris** This dataset has 150 entries with 4 features and 1 ternary label. Ran in the reproduction with 3 clusters. N = 150
5. **Breast Cancer Wisconsin (Diagnostic)** [4] This dataset has 569 entries with 30 features and 1 binary label. We use 3 clusters in the replication.
6. **Pima Indians Diabetes Database** [5] This dataset has 768 entries with 8 features and 1 binary label. We used 3 clusters in the replication. The data was normalized to be in the range [0, 1].

   Note that the number of clusters depend on the latent-space representation, and thus, are user dependent.

4.3 Hyperparameters

Tensorflow [1] Experiments For the reproduction of the original experiments, we use the same hyperparameters as the original authors.

Pytorch For our implementation of the scVIS model, we use l2 regularization of 0.001, learning rate 0.01, and perplexity of 10. Furthermore, the degree-of-freedom for the studentT distribution is set to 2.0. Perplexity and the degree-of-freedom is used same as the original scVIS implementation. We train the scVIS model for 200 epochs, and use validation set to monitor the training process. For training the TGT explanations, we closely follow the settings from Plumb et al. [7]. We initialize the deltas as zero vectors. The regularization parameter λ tuned by grid search over a fixed range $[0.0, 5.0]$ incremented by 0.5. Defining the metrics for TGT requires careful setting of the ϵ hyper-parameter. We follow the *self-similarity* condition (transformations of clusters to themselves should, theoretically, have correctness and coverage to be 1.0), and increase the ϵ in the range $[0.0, 2.0]$ with increments of 0.02 until the correctness and coverage metrics are greater than 0.95. Furthermore, we use the truncation values(TV)(refer Table 1) to evaluate on the sparsity of the explanations. For the diabetes and breast cancer dataset, we use the same truncation values as for Housing.

Dataset	**Truncation Values(TV)**	ϵ
Single Cell RNA	50, 100, 250, 500, 1000, 15000	0.75
Heart Disease	1, 3, 5, 7, 9, 11, 13	1.0
Housing	1, 3, 5, 7, 9, 11, 13	1.5
Iris	1, 2, 3, 4	0.75

Table 1: Truncation Values (TV) and ϵ value used for each of the dataset.

4.4 Experimental setup and code

We closely follow the experimental setup in the original paper for our experiments. We make our Pytorch code available [6] to further support the reproducible research. We reran the code of the original authors with new clustering models and

[4] https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

[5] https://www.kaggle.com/uciml/pima-indians-diabetes-database

[6] https://github.com/elfrink1/FACT

Explanation	x_1	x_2	x_3	x_4
$0 \rightarrow 1$	-1.01	-0.02	0.00	-0.88
$0 \rightarrow 1$	-1.05	0.99	0.00	-0.88
$0 \rightarrow 3$	0.00	0.89	0.00	0.00

Table 2: Explanations for the synthetic dataset as given by our implementation.

new deltas. We optimize the compressed-sensing based objective function for the TGT algorithm using the gradient descent algorithm. Our scaling extension is easily integrated in the source code, and can be optimized in a similar way. We train the scVIS models on the Lisa computing cluster [7]. We use approximately 30 hours of GPU time. We train the TGT explanations on CPU (Intel i5).

5 Results

For the reproduction of the authors' experiments, we achieve approximately similar results to the original paper. The TGT method does seem to outperform DBM method. The TGT explanations also have high similarity across sparsity levels. Furthermore, the TGT algorithm is able to identify known causal structure in synthetic data. However, we are unable to match the results on the modified and corrupted data to a good precision. We describe the results in the following sections:

5.1 Results reproducing original paper

5.1.1 Coverage, Correctness and similarity

In figure 1 we can see a comparison between the correctness, coverage and similarity of the TGT and DBM methods. Note that the DBM always has similarity 1. The similarity of TGT stays between 1 and 0.9, which supports claim 2.

We see that the coverage and correctness are similar for the heart disease dataset. On the Iris dataset, the coverage is comparable but the correctness is better for TGT. In both housing and RNA, the coverage and correctness are better at less features and similar for more features. Overall, these results support claim 1, especially for a small amount of features.

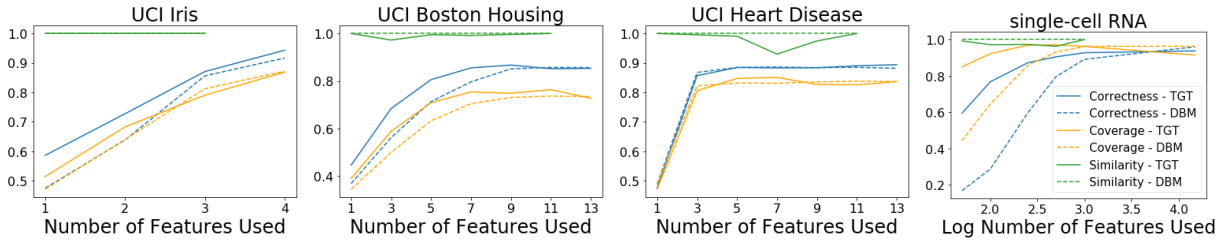

Figure 1: Reproduced TGT metrics

5.2 Explaining Causal Structure in the Synthetic Data

We verify the claim that TGT identifies the causal structure in the data (claim 3). The synthetic dataset is generated same as the original paper, i.e. $x_1, x_2 \sim \mathcal{N}(0, 0.2) + \text{Bern}(0.5)$, $x_2 \sim \mathcal{N}(0, 0.05)$, $x_4 \sim x_1 + \mathcal{N}(0, 0.05)$. Note that this dataset has four different clusters, caused by the first two dimensions x_1 and x_2, x_3 is noise, and x_4 is correlated with x_1 and x_2. The authors claim that for this synthetic data, TGT is able to find that x_4 is not the cause for any group. However, the said claim cannot be re-verified. Interestingly, the re-run of their code doesn't provide the justification either to the degree as mentioned in the paper. We observe that both TGT and DBM are able to identify x_3 is not causing any groups. Thus, in this scenario, both TGT and DBM are comparable. Refer table 2 for the explanations obtained. We, hereby note, that the explanations vary across multiple runs. However, the values across the third dimension are consistently approximately 0.

[7] https://userinfo.surfsara.nl/systems/lisa

5.2.1 Feature modifications

For each of the UCI datasets, the original authors add a 'corrupted' version where an extra cluster is added with artificial feature modification. With the exception of the modified features, the corrupted class is a copy of a chosen target class. They train TGT explanations using both the original scVIS model for the respective dataset and a model retrained on the corrupted dataset. We reproduce these experiments to see if TGT correctly attributes the difference between the target and corrupted class to the right features. Refer to Appendix A.1 for the illustrated figures.

For the heart disease dataset, features 6 and 8 are modified but TGT identifies those differences to be in features $2, 5$ and 10. This does not support claim 4. When training TGT on the scVIS model which was trained on the corrupted data, it is able to identify the correct features.

For the housing dataset, features 1 and 9 are modified. TGT clearly identifies feature 1 and also identifies feature 9 but not by the right significance. This does support claim 4. On the retrained scVIS model, the correct features are identified

For the Iris dataset, feature 1 was modified, which is correctly identified by TGT on both the original and retrained scVIS model. This supports claim 4.

5.3 Results beyond original paper

5.3.1 PyTorch replication

We replicated the TGT algorithm in PyTorch, since the FACT-AI course at the UvA requires this. Since the scVIS model is not available in PyTorch, we had to implement a clustering method similar to this. Since the TGT method should be model agnostic, this should not be an obstacle. Since the clusters in the Tensorflow reproduction had to be manually selected, we decided to the Scikit learn [8] kmeans implementation for our cluster selection. Due to differences in the clustering model and cluster selection, we cannot directly compare the coverage and correctness metrics between our Pytorch replication and the TensorFlow reproduction.

5.3.2 UCI Heart Disease and Housing

In figure 2 we see the results of our replication on the UCI Boston Housing and heart disease datasets. For the housing data, the TGT method seems to slightly outperform DBM both with and without scaling. This supports claim 1. The deltas and logit gammas show high similarity, supporting claim 2. For the heart data we do not see a difference in performance without scaling while TGT with scaling performs slightly worse.

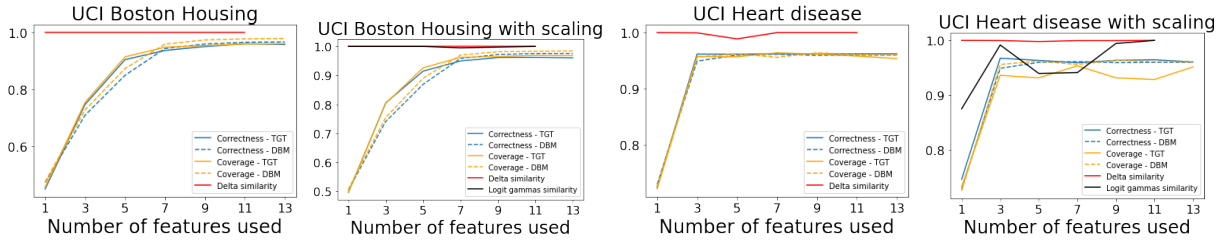

Figure 2: PyTorch replication of UCI Housing Heart disease data

5.3.3 Breast Cancer Wisconsin (Diagnostic) and Pima Indians Diabetes Database

In figure 3 we see the results for the diabetes and breast cancer dataset. For the diabetes dataset, TGT with and without scaling outperforms DBM when more than one feature is used. This supports claim 1. Since the delta similarity is close to 1, claim 2 is also supported. For the heart disease dataset, we see similar performance for DBM and TGT and slightly worse performance with scaling. The deltas still have high similarity, supporting claim 2.

[8] https://scikit-learn.org/stable/index.html

	0 to 1		1 to 0		δ_1	δ_2	δ_3	δ_4	γ_1	γ_2	γ_3	γ_4
	Cr	*Cv*	*Cr*	*Cv*								
TGT	1	0.529	0.333	1.000	2.581	0.014	0.001	0.842	-	-	-	-
Scaling	1	0.529	0.520	1.000	2.608	0.023	-0.001	0.927	0.879	0.009	0.002	0.007

Table 3: The deltas and gammas for the mapping from group 0 to group 1 on the modified synthetic dataset for regular TGT and TGT with scaling. *Cr* and *Cv* indicate coverage and correctness, respectively.

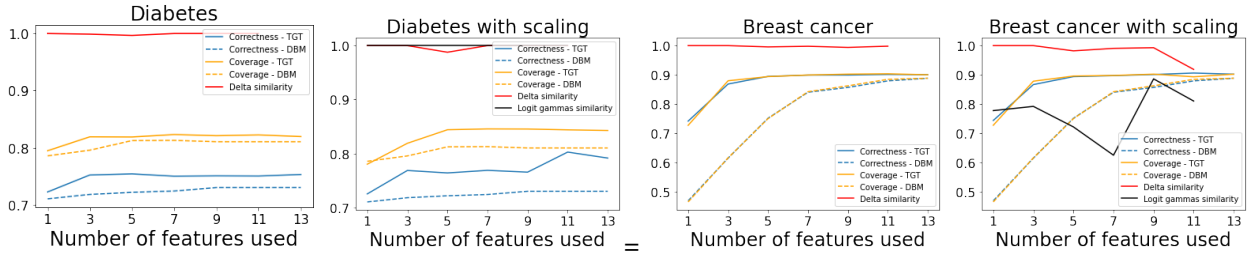

Figure 3: PyTorch replication on diabetes and breast cancer data

5.3.4 Scaling extension

The original authors use translations as their explanation functions. We wondered if these transformation are expressive enough. We replicated our experiment on the housing, heart disease, diabetes and breast cancer datasets. Besides the δ parameters, each cluster now has a γ parameters. The transformation from cluster 0 to i is now given by: $t_{0\rightarrow i} = e^{\gamma_i} \odot x + \delta_i$ The gammas are truncated just like the deltas and their $L1$ norm is added to the regularization term. Note that these transformations are strictly more expressive. If γ is the zero vector, these transformations reduce to regular TGT.

In figure 2 we can see the difference in performance on two datasets included in the original experiments. Scaling does not seem to improve performance on the housing dataset and slightly decreases performance on the heart disease dataset. The similarity of the gammas is mostly above 0.9.

In figure 3 we see the same metrics plotted for the two new datasets that we added. For the diabetes dataset, the performance improves slightly and the gammas show high similarity. For the breast cancer dataset, the performance is about the same but the gammas show relatively low similarity.

Altogether, these results suggest that the addition of scaling does not significantly improve the accuracy and correctness while making the transformations more complex. We do not recommend the addition of scaling in the explanation functions.

5.3.5 Experiment with Modified Synthetic Data

In order to study the efficacy of the proposed scaling function, we perform experiments on the synthetic dataset. We modify one of the groups of points G by performing the operation $ax_i^j + b$, where i corresponds to the group number and j denotes the feature dimension which we modify. We define $a \sim \mathcal{U}(1.0, 2.0)$ and $b \sim \mathcal{U}(-0.5, 1.0)$. We add modified group G' into the original data D to get the new data $D^{'}$. We follow the experiment setup from the original paper as: a) $r(G^{'})$ should form a different group of its own. b) $G^{'}$ should be within the distribution of the original D. In this study, we want to investigate whether the TGT with scaling is able to recover the modifications and if in doing so it affects the explanations between other groups. We observe that the explanations with scaling are able to recover the modification to an approximate degree, and give better correctness as compared to the regular TGT. Interestingly, the translations explanations of the scaling extension are approximately equal to the deltas of the regular TGT. The exact results can be found in Table 3. Figures 8 and 9 in the appendix show the data spread and resulting translations.

5.3.6 Experiments with Probing Classifier

To further investigate the efficacy of TGT explanations, we use a probing-classifier [2] as a proxy to study the qualitative differences of the features selected by TGT and DBM. For each cluster, we train a binary classifier with features ranked highest by TGT and DBM at different sparsity levels K. We compute the overall accuracy at each sparsity level using the ensemble of these binary classifiers. As can be seen in Figure 4, the results demonstrate that for sparser explanations,

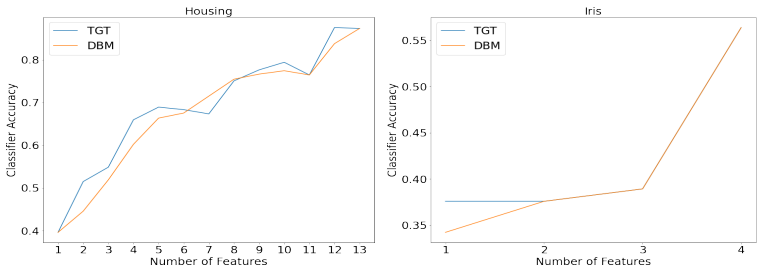

Figure 4: Classification accuracy of probing classifier at different sparsity levels for Housing (left) and Iris (right) dataset.

TGT selects features that lead to higher accuracy of the ensemble classifier than those selected by DBM. This further validates the paper's claim that TGT leads to better sparse explanations as compared to DBM. Furthermore, we also use the probing classifier to understand the differences between the groups. For each pair of group, we train a Binary Linear Classifier to predict the group of a test point. We, then, investigate the feature importances of the classifier towards decision making. We ascertain that the features classifier give more importance to while decision making are the defining property of the class. Interestingly, we find that the more important features according to the classifier correspond to the explanations provided by the TGT algorithm. Refer to A.3. This provides further evidence that TGT is able to find real distinctive signals as explanations.

6 Discussion

Based on the reproduction of the original experiments. Claims 1 and 2 seem to hold, the experiments for claim 4 do not all support it, but the claim does seem to hold. Claim 1 and 2 seem to hold in particular for sparse explanations. The evidence for claim 3 is inconclusive. The coverage and correctness in our reproduction were not always the same as in the original paper. It is difficult to compare these metrics for different clustering outcomes, as they depend on the ϵ parameter which depends on the clustering.

A major difficulty in reproduction is the cluster selection. When retraining the scVIS model, the latent space representation structure changes. The authors provide no method as to determine the different clusters other than visual inspection. Cluster selection could be an explanation for the differences in results between the original experiments and our reproduction. To verify the results with more confidence, a robust method for cluster selection might be required.

6.1 What was difficult

The first difficulty that we encountered was the finding of the right environment. Their code depended on deprecated functionality and could thus not be used with the latest software versions. The use of python notebooks also introduced some difficulty when attempting to use these for training on a computing cluster. This is because computational jobs have to be scheduled and there is no intermediate interaction. The python notebooks have to be translated into a python script.

Another difficulty was the selection of clusters. When retraining the scVIS, we get a different latent space representation. The clusters have to be manually selected. This also means there does not have a be a correspondence between the ordering of clusters using the original latent space representation and clusters in the retrained one. Because of this, we cannot compare the metrics between our reproduction and the original experiments.

6.2 Communication with original authors

We asked the original authors for clarification on how to choose the ϵ hyper-parameter. However, it became apparent that we had misread, and the procedure is indeed adequately reported in the paper.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Yonatan Belinkov, Sebastian Gehrmann, and Ellie Pavlick. Interpretability and analysis in neural NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 1–5, Online, July 2020. Association for Computational Linguistics.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019.

[5] Jiarui Ding, Anne Condon, and Sohrab P. Shah. Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *bioRxiv*, 2017.

[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[7] Gregory Plumb, Jonathan Terhorst, Sriram Sankararaman, and Ameet Talwalkar. Explaining groups of points in low-dimensional representations, 2020.

[8] Karthik Shekhar, Sylvain W Lapan, Irene E Whitney, Nicholas M Tran, Evan Macosko, Monika Kowalczyk, Xian Adiconis, Joshua Z Levin, James Nemesh, Melissa Goldman, Steven Mccarroll, Constance L Cepko, Aviv Regev, and Joshua R Sanes. Comprehensive classification of retinal bipolar neurons by single-cell transcriptomics. *Cell*, 166:1308–1323.e30, 08 2016.

[9] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[10] Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review, 2020.

[11] Haozhe Xie, Jie Li, and Hanqing Xue. A survey of dimensionality reduction techniques based on random projection, 2018.

A Extra figures

A.1 Corrupted datasets

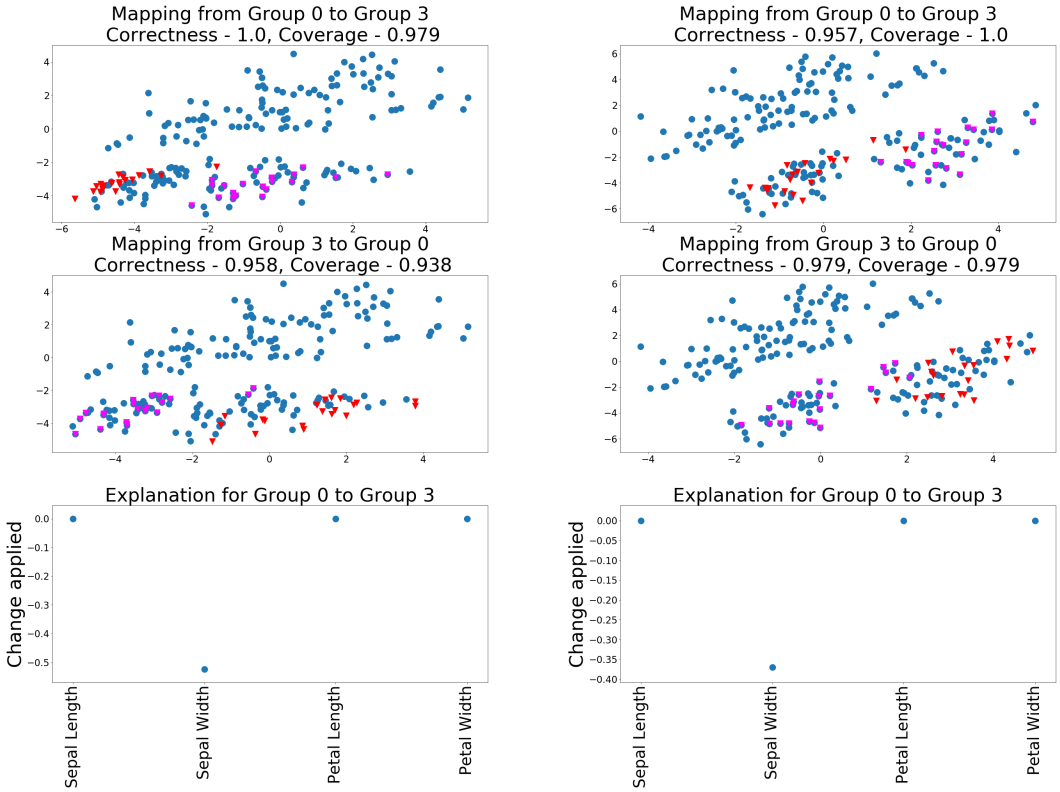

Figure 5: Explanation for corrupted features on UCI Iris

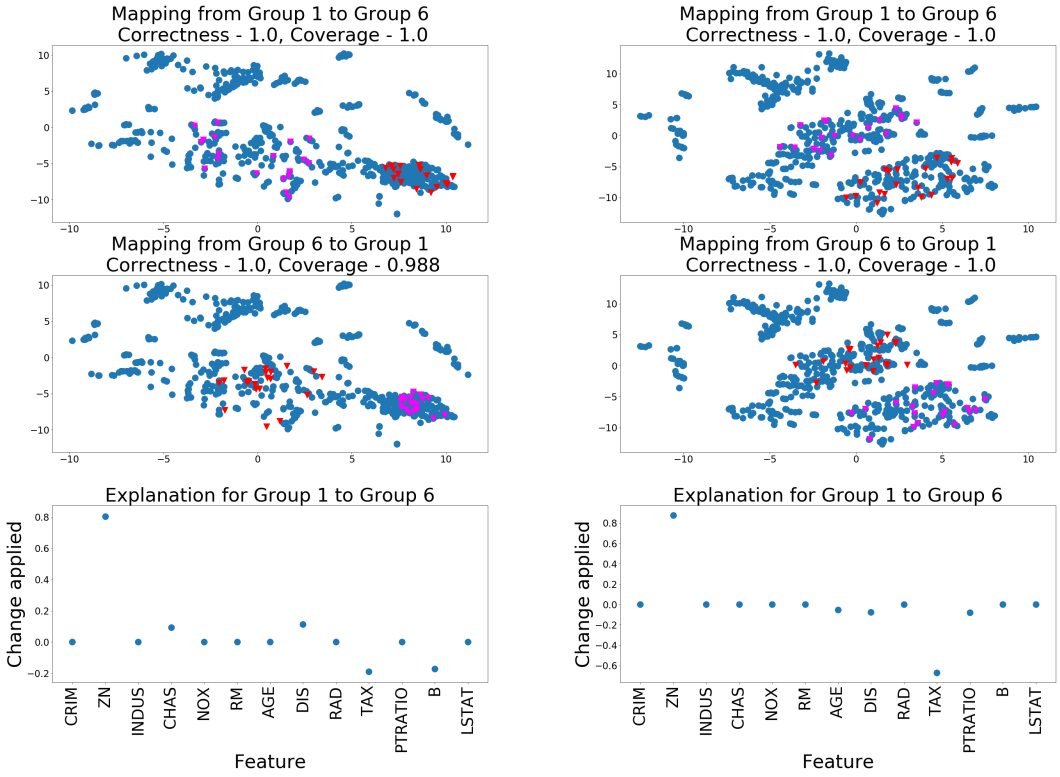

Figure 6: Explanation for corrupted features on UCI Boston Housing

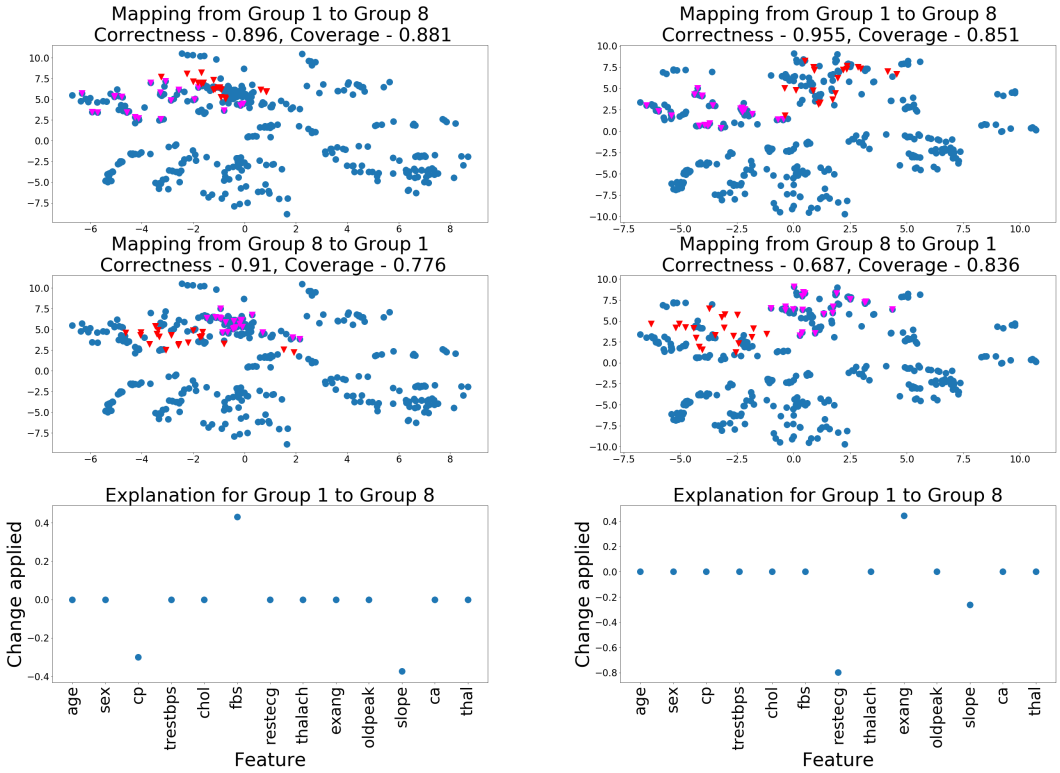

Figure 7: Explanation for corrupted features on UCI heart disease

A.2 Synthetic data

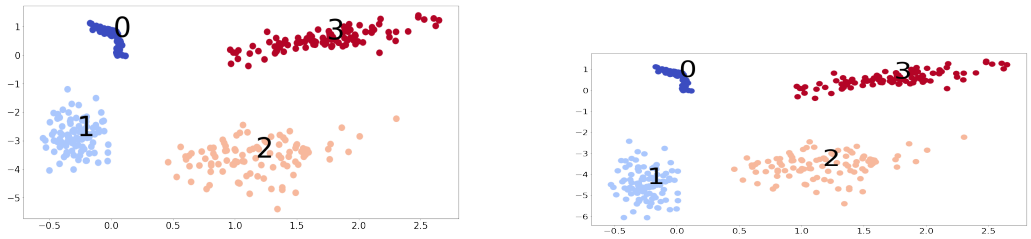

Figure 8: a) Synthetic data b) Synthetic data with the modification applied. We modify the data from group 1 across the 0th dimension by $ax_1^0 + b$. Here a and b are 2.0, 0.60 respectively.

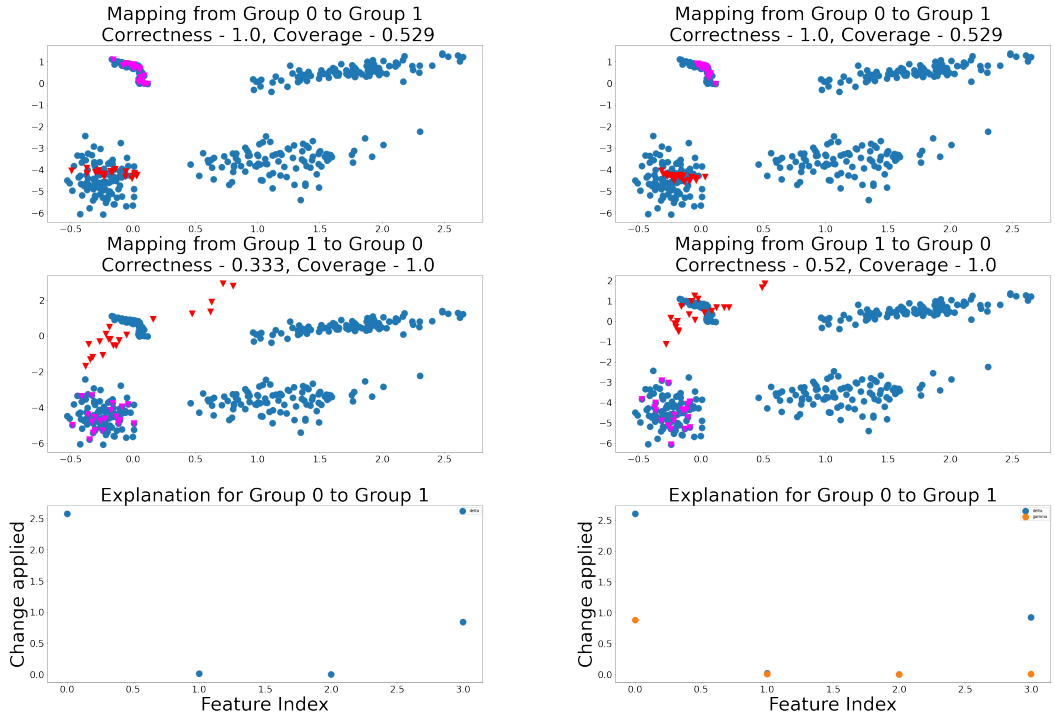

Figure 9: We compare the explanations from the TGT algorithm (left) and the TGT with scaling extension algorithm(right) on the modified synthetic data. We can observe that the TGT with scaling extension has better correctness, and is able to identify the scaling we have applied across the 0th dimensionality. The γ for this dimension is 0.87, which means the scaling factor is $e^{\gamma} \approx 2.38$. Moreover, the translation parameters are approximately same in both the variants of the TGT.

A.3 Probing Classifier and Feature Importance

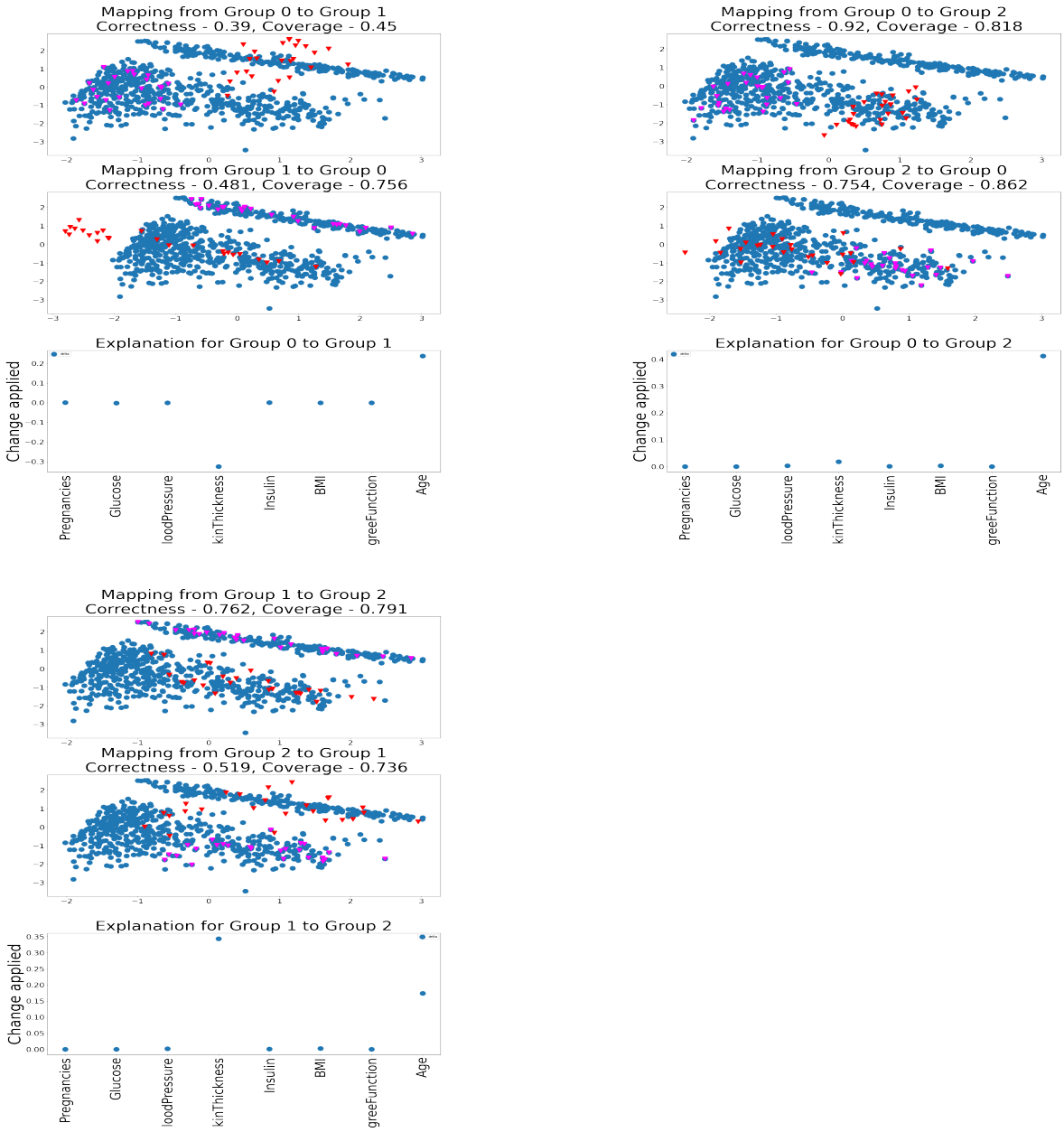

Figure 10: Explanations between different groups for the Pima Indians Diabetes Database.

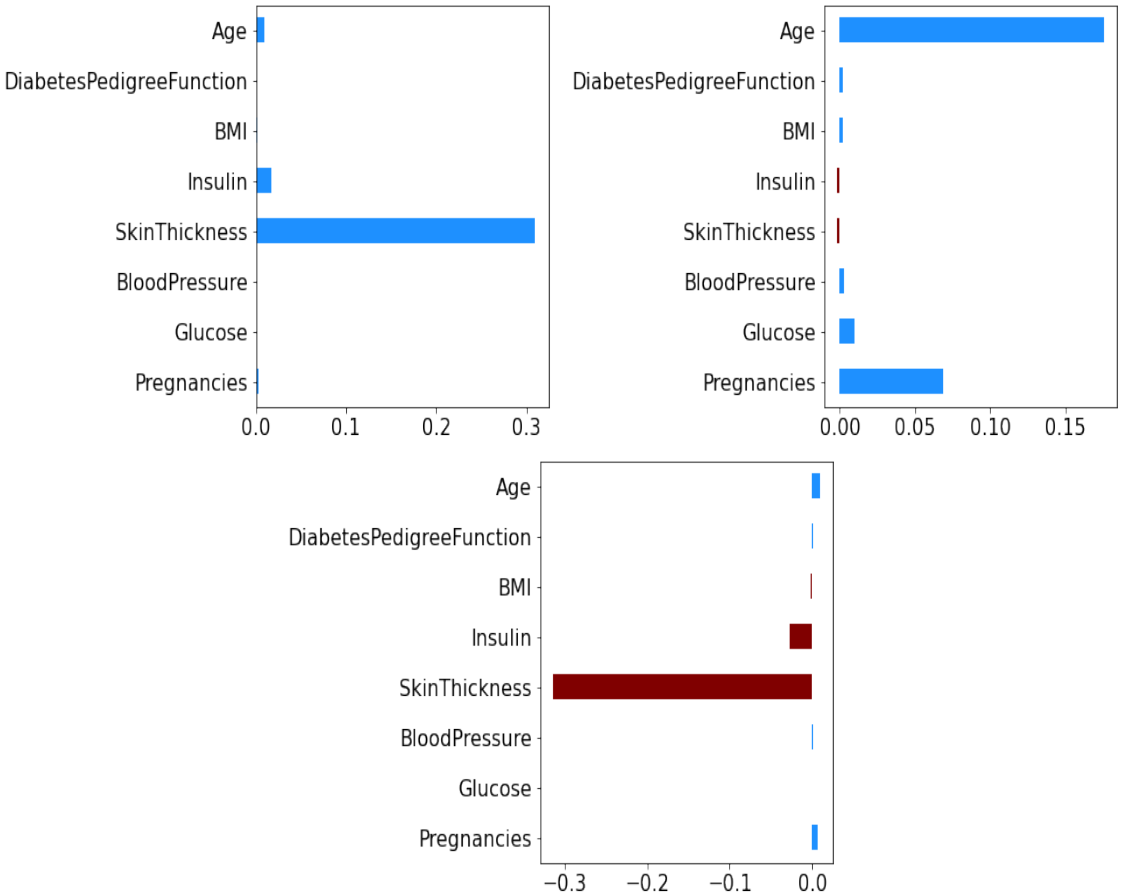

Figure 11: Feature Importance by the binary classifier for the Pima Indians Diabetes Database. a) (top-left): Feature importance for the classifier between groups 0 and 1. b) (top-right): Feature importance for the classifier between groups 0 and 2. c) (bottom-left): Feature importance for the classifier between groups 1 and 2. We note that the classifiers give significant feature importances to the features which correspond to the deltas (refer fig. 10).