Counterfactual Explanations for Student Outcome Prediction with Moodle Footprints

Anjana Wijekoon, Nirmalie Wiratunga, Ikechukwu Nkisi-Orji, Kyle Martin, Chamath Palihawadana, and David Corsar

School of Computing, Robert Gordon University, Aberdeen AB10 7GJ, Scotland, UK
{a.wijekoon, n.wiratunga, i.nkisi-orji, k.martin3, c.palihawadana, d.corsar1}@rgu.ac.uk

Abstract. Counterfactual explanations focus on "actionable knowledge" to help end-users understand how a machine learning outcome could be changed to one that is more desirable. For this purpose a counterfactual explainer needs to be able to reason with similarity knowledge in order to discover input dependencies that relate to outcome changes. Identifying the minimum subset of feature changes to action a change in the decision is an interesting challenge for counterfactual explainers. In this paper we show how feature relevance based explainers (such as LIME), can be combined with a counterfactual explainer to identify the minimum subset of "actionable features". We demonstrate our hybrid approach on a real-world use case on student outcome prediction using data from the CampusMoodle Virtual Learning environment. Our preliminary results demonstrate that counterfactual feature weighting to be a viable strategy that should be adopted to minimise the number of actionable changes.

Keywords: Explainable AI, Counterfactual, LIME

1 Introduction

Understanding a user's explanation need is central to a system's capability of provisioning an explanation which satisfies that need [4]. Typically an explanation generated by an AI system is considered to convey the internal state or workings of an algorithm that resulted in the system's decision [6]. In machine learning (ML) the decision tends to be a discrete label or class (or in the case of regression tasks a numeric value). Although explanations focused on the internal state or logic of the algorithm is helpful to ML researchers it is arguably less useful to an end-user who may be more interested in what in their current circumstances could be changed to receive a desired (better) outcome in the future. This calls for explanations that focus on discovering relationships between the input dependencies that led to the systems' decision.

A Nearest-Like Neighbours (NLN) based explainer, could focus on input dependencies by identifying similarity relationships between the current problem

and the retrieved nearest neighbour [3]. Research has shown that when similarity computation is focused on feature selection and weighting, it can significantly improve retrieval of NLN [8, 7]. Accordingly it would be reasonable to expect that NLN-based explanation generation would also benefit from the knowledge of feature importance. Certainly having to focus on a few key features in domains with large numbers of features is likely to improve the cognitive burden of understanding NLN-based explanations.

Unlike NLN based explanations, a counterfactual explanation focuses on identifying "actionable knowledge"; which is knowledge about important causal dependencies between the input and the ML decision. Such knowledge helps to understand what could be changed in the input to achieve a preferred (desired) decision outcome. Typically a Nearest-Unlike-Neighbours (NUN) is used to identify the number of differences between the input and its neighbour, that when changed can lead to a change in the system's decision [2]. A key challenge that we address in this paper is to identify the minimum subset of feature value changes to achieve a change in the decision - the "actionable features". In this paper, we discover actionable knowledge as the minimum number of feature changes using feature relevance based explainer methods like LIME [5].

The rest of the paper is organised as follows. Section 2 presents the Moodle[1] use case followed by Section 3 which presents the proposed methods to improve the discovery of actionable features. Section 4 presents the Moodle dataset details, evaluation methodology, performance metrics and results. Finally we draw conclusions and discuss future work in Section 5.

2 Use Case: Student Outcome Prediction

Student information systems are increasingly using ML to identify and predict events in a student's journey to help improve early intervention and mitigate at-risk students [1]. They help both the student to adopt positive behaviours and to pinpoint improvements to teaching style and content. We have used the CampusMoodle Virtual Learning Environment (VLE) to construct a student footprint dataset for a single module of study delivered within Robert Gordon University in Semester 1 of 2020/2021. VLE interactions help to capture vital touchpoints that can be used as proxy measures of student engagement. It contains a wealth of relevant data for each student such as the number of times they have accessed individual learning materials gathered over a period of ten weeks (full details of the dataset are described in Section 4.1).

The use case dataset links the "CampusMoodle footprint" with the "Assessment Outcomes" data. This creates a classification task to predict the likely final grade for a given student, and an associated explanation task to explain this outcome given the student's Moodle footprint. A student may want to know why they received a lower grade and in response the nearest neighbour instance could be used to factually confirm the credibility of that lower grade i.e. "other students like you also received a similar lower grade". Although such precedence can

[1] Learning management system https://moodle.org/

be used to justify the student's circumstances, it is not as useful as a "counterfactual" explanation, which can provide indications as to how another student with a similar profile ended up with a better grade simply because they had completed a formative quiz exercise. Here we want the explainer to identify a few important features to consider, instead of expecting the end-user to analyse hundreds of features used to describe the Moodle footprint.

3 Actionable Explanations

Given a query instance, $x = \{x_1, x_2, ..., x_m\}$, its counterfactual, $\hat{x} = \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_m\}$, is identified as the nearest-unlike-neighbour (NUN) in the Euclidean feature space [2]. Here m is the number of features used to represent instances (i.e. x_i and $\hat{x}_i$ pairs).

$$d(x, \hat{x}) = \sqrt{\sum_{i=1}^{m}(x_i - \hat{x}_i)} \tag{1}$$

The goal of a counterfactual explanation is to guide the end-user to achieve class change (i.e. actionable), with a focus on minimising the number of changes needed to flip the decision to a more desirable outcome for the end-user. Therefore in order for an instance to qualify as a NUN, for a given query instance, it needs to be the nearest neighbour with a different class to that of the query. Discovering the minimal number of actionable features (from a maximum of m potential feature changes) and minimising the amount of change required on each actionable feature are two main challenges for counterfactual explainers.

3.1 Actionable Feature Discovery with LIME

LIME [5] is a model-agnostic explainer which creates an interpretable model around a query, x. The resulting local surrogate model is interpretable and only locally faithful to the classifier but not globally. LIME creates a set of perturbations within the neighbourhood of x. The original classifier is used to obtain the class labels for perturbed data instances and this new labelled dataset is used to create a local interpretable model (often a linear model). The new interpretable model is used to predict the classification outcome of x that needs to be explained. Accordingly, LIME outputs how each feature x_i contributed to the final classification outcome.

These local feature contribution weights form the main LIME output, which is a list of tuples (w_i, i), where each feature, x_i, has its corresponding weight.

$$\texttt{LIME output} = [(w_1, 1), (w_2, 2), ..., (w_m, m)] \tag{2}$$

A positive weight ($w_i >= 0$) indicates that the corresponding feature contributes positively and a negative weight ($w_i < 0$) corresponds negatively towards the predicted class. Figure 1 shows a LIME explanation for a Moodle data instance

which predicted a *Lower* grade (See Section 4.1 for more details on the dataset). Weights are ordered such that highest weights appear first, regardless of a positive or negative contribution to the predicted class. Here we can see that the

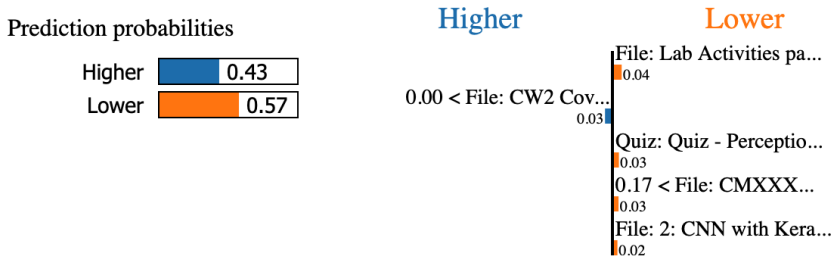


Fig. 1: LIME explanation for a student who received a *Lower* grade from the prediction model

feature "Study Area: CM4107" had the highest positive impact and feature "Assignment: DROPBOX WRITEUP: CW2" had the highest negative impact on the *Lower* grade classification.

LIME can be used to provide relevance weights for both the query and its potential NUN. The number of feature changes, n, required to achieve a class change, can range from 1 to m ($1 <= n <= m$). We propose two methods to find these actionable features, with the goal of minimising the number of feature changes (n) needed for a succinct yet actionable explanation. The first method is to replace the values of the most important features in the query with the corresponding feature values from its NUN; or a second alternative is to identify the most relevant features of the NUN and reuse those feature values in the modified query instance.

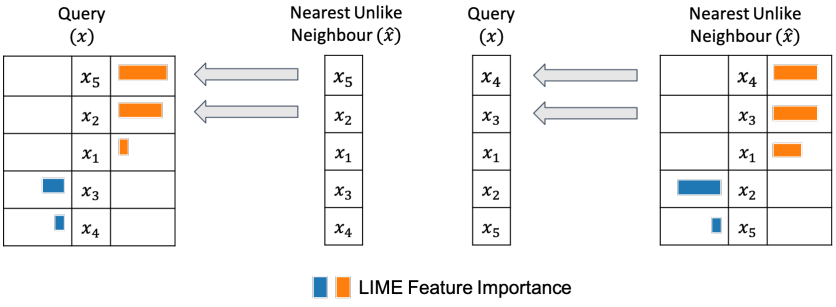


Fig. 2: Using LIME feature importance of query (Left) or NUN (right) to discover actionable features in Counterfactuals

We illustrate these two methods in Figure 2. Here, we depict an example with 5 features where we replace features on the query until a class change is observed. In the left, the query features are ordered by their LIME feature weights and the most significant features are replaced by the respective NUN feature values (Method 1). In the right, the NUN features are ordered by their LIME feature weights and the most significant features are reused by the query (Method 2). Method 1 and 2 achieve class change with 3 and 2 feature replacements respectively.

The general functions for actionable feature discovery using counterfactuals is shown in Algorithms 1 and 2. Here $\mathcal{F}(x) = y$ is the classifier prediction for the query and, $\hat{y} = \mathcal{F}(\hat{x})$ is the class prediction for the NUN. Given an instance, an Actionable Feature Ordering function, AFOrder$_{LIME}(.)$ returns the feature indices, $\mathcal{I}$, ordered by the local importance of a given instance's features for the resultant class prediction. Thereafter Actionable Feature Discovery function, AFDiscovery, uses $\mathcal{I}$ to create a perturbed version of the query, x', where important features are replaced one at a time with the corresponding feature values from the NUN. The features with equal values are skipped. Feature values are iteratively replaced until the actionable change condition is met i.e. $\mathcal{F}(x') = y' \wedge y \neq y'$. Clearly the fewer replacement iterations needed the better the actionable features being discovered. Crucial to this minimisation is the ordering methods that influence $\mathcal{I}$:

Method1: uses $AFOrder_{LIME}(x)$ returning indices ordered on the basis of the feature importance that led to the class prediction of the query; and

Method2: uses $AFOrder_{LIME}(\hat{x})$ returning indices ordered on the basis of feature importance that led to the class prediction of the NUN.

Algorithm 1 AFDiscovery

Require: (x, y): query and label pair
Require: $(\hat{x}, \hat{y})$: NUN and label pair
Require: $\mathcal{F}$: classification model
Require: $\mathcal{I}$: list of feature indices
Declare $y' = y$; $x' = x$; $n = 0$
while $y' = y$ **do**
    $x'[\mathcal{I}[n]] = \hat{x}[\mathcal{I}[n]]$
    $y' = \mathcal{F}(x')$
    increment n
end while
return n

Algorithm 2 AFOrder$_{LIME}$

Require: q: Data instance for which LIME weights are retrieved
$[(w_1, 1), (w_2, 2), ..., (w_m, m)] = LIME(q)$
return $\mathcal{I}$: list of feature indices ordered by w

4 Evaluation

The goal of this evaluation is to determine the effectiveness of the actionable feature discovery algorithm with different ordering strategies. We compare the query and NUN feature relevance based ordering over a random ordering of

features for actionable feature discovery. The following algorithms are compared in this evaluation:

1. **Random**: Instead of AFOrder$_{LIME}$, Randomly ordered set of feature indices is used in AFDiscovery.
2. **Method1**
3. **Method2**

4.1 Moodle Dataset

Moodle dataset consists of Moodle footprint of 79 students who were enrolled for a course during September and December 2020 at RGU. There are 95 features where each feature refers to the number of times the resource on the Course page was accessed by a student. For instance, feature "Assignment: DROPBOX WRITEUP: CW1" refers to the submission dropbox provided for submitting coursework 1 and feature value refers to the number of times the student accessed the dropbox.

Predicting student outcome using Moodle footprint The ML task of this dataset is to predict if a student gets a higher or a lower grade based on their Moodle footprint. We consider grades *A* and *B* as *Higher* grades and *C*, *D*, *E* and *F* as *Lower* grades. Grades were consolidated as *Higher* and *Lower* to mitigate the comparably lower number of data instances and class imbalance. Five outlier student cases were also filtered out using the Density-based Spatial Clustering method offered in sklearn Python libraries. This formed a dataset of 74 instances.

A RandomForest classifier was used to predict the grade based on the Moodle footprint. There were 500 trees created using Python sklearn libraries which were selected after an exhaustive search for the best ML algorithm using an AutoML method. The classifier achieved 83% accuracy over three stratified folds. Note that for our results when explaining an outcome, we assume that the classifier has correctly predicted the grade.

Explaining predicted student outcome There are two explanation intents explored in this paper: firstly a counterfactual type question, *Why* student A did *not* receive a grade X?; or an alternative question *Why* did student A receive grade Y? The latter can be explained using the LIME explainer presenting the contribution of the most important features for the predicted grade; and the former *Why not* type question explained through a counterfactual explanation formed using methods for actionable feature discovery.

4.2 Evaluation Methodology

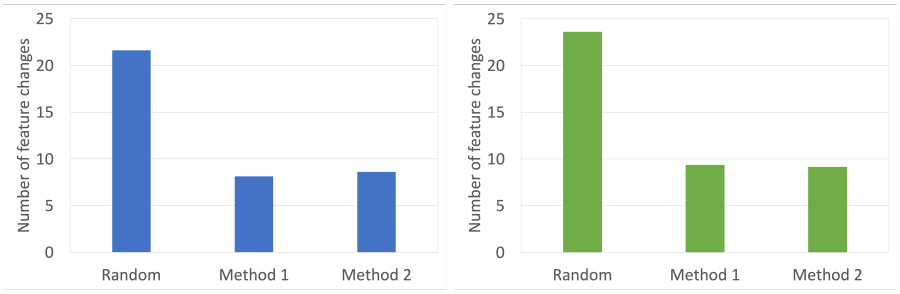
We consider two versions of the Moodle dataset in this evaluation: First with all 74 instances each being considered as the query; and Second with 50 instances

from the class *Lower* grade as the query. For each query the number of feature changes required to observe a class change is calculated using the three methods described above. Final value is computed as the mean number of actionable features. The first dataset provides a generalised comparison between the performances of the three methods. The second dataset represents the situation that is most likely to require a counterfactual explanation, i.e. students who ask how to get a *Higher* grade or why did they not receive a *Higher* grade. Accordingly the second version will further verify the comparative performances of the three methods in a domain-specific context.

4.3 Results

Figures 3a and 3b present the comparison of the three methods for discovering actionable features using the two datasets. With the full dataset, Random, Method 1 and Method 2 observe a class change with 21.62, 8.14 and 8.61 feature changes on average. Similar results are observed in the domain-specific case where Random, Method1 and Method2 observe a class change with 23.57, 9.34, 9.13 feature changes on average. Accordingly, two methods proposed in this paper has significantly outperformed the random feature ordering approach to achieving class change. LIME has highlighted the features that are actionable, thus minimising the number of changes required.

Furthermore, with both datasets, Method1 and Method2 performances are comparable. Accordingly, reusing features of NUN ordered by the significance of how each contributed to predicting class label $\hat{y}$ is found to be as effective as considering Query feature significance. Given that a student's goal is to find actionable resources on Moodle towards achieving a different grade (commonly a *Higher* grade) we find these results align with our domain knowledge.



(a) All Data (b) Data labelled *Lower Grade*

Fig. 3: Comparative Results

5 Conclusion

In this paper, we presented a novel approach to finding actionable knowledge when constructing an explanation using a counterfactual. We used feature relevance explainer LIME to order features that are most significant to a predicted class and then used that knowledge to discover the minimal actionable features to achieve class change. We demonstrated our approach using a dataset from the CampusMoodle VLE where a course outcome (i.e. grade) is predicted using the student's Moodle footprint. Our empirical results showed that our approach is significantly effective when compared with the random approach to discovering actionable features. In future, we plan to extend our approach to use other feature relevance algorithms and further minimise the number of features to create succinct actionable explanations.

Acknowledgements This research is funded by the iSee project (https://isee4xai.com/) which received funding from EPSRC under the grant number EP/V061755/1. iSee is part of the CHIST-ERA pathfinder programme for European coordinated research on future and emerging information and communication technologies.

References

1. Conijn, R., Snijders, C., Kleingeld, A., Matzat, U.: Predicting student performance from lms data: A comparison of 17 blended courses using moodle lms. IEEE Transactions on Learning Technologies **10**(1), 17–29 (2016)
2. Keane, M.T., Smyth, B.: Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable ai (xai). In: International Conference on Case-Based Reasoning. pp. 163–178. Springer (2020)
3. Kenny, E.M., Keane, M.T.: Twin-systems to explain artificial neural networks using case-based reasoning: Comparative tests of feature-weighting methods in ann-cbr twins for xai. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. pp. 2708–2715. International Joint Conferences on Artificial Intelligence Organization (7 2019). https://doi.org/10.24963/ijcai.2019/376, https://doi.org/10.24963/ijcai.2019/376
4. Mohseni, S., Zarei, N., Ragan, E.D.: A survey of evaluation methods and measures for interpretable machine learning. arXiv preprint arXiv:1811.11839 (2018)
5. Ribeiro, M.T., Singh, S., Guestrin, C.: " why should i trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1135–1144 (2016)
6. Wachter, S., Mittelstadt, B., Russell, C.: Counterfactual explanations without opening the black box: Automated decisions and the gdpr. Harv. JL & Tech. **31**, 841 (2017)
7. Wettschereck, D., Aha, D.W., Mohri, T.: A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. Artificial Intelligence Review **11**(1), 273–314 (1997)
8. Wiratunga, N., Koychev, I., Massie, S.: Feature selection and generalisation for retrieval of textual cases. In: European Conference on Case-Based Reasoning. pp. 806–820. Springer (2004)