# Contextual Importance and Utility in R: the ‘ciu’ Package

**Kary Främling**[1][2]
[1] Department of Computing Science, Umeå University, Mit-huset, 901 87 Umeå, Sweden
[2] Department of Computer Science, Aalto University, Konemiehentie 1, 02150 Espoo, Finland
kary.framling@umu.se, kary.framling@aalto.fi

**Abstract**

Contextual Importance and Utility (CIU) are concepts for outcome explanation of any regression or classification model. CIU is model-agnostic, produces explanations without building any intermediate interpretable model and supports explanations on any level of abstraction. CIU is also theoretically simple and it is computation-efficient. However, the lack of CIU software that is easy to install and use appears to be a main obstacle for the interest in and adoption of CIU by the Explainable AI community. The *ciu* package in R that is described in this paper is intended to overcome that obstacle at least for problems involving tabular data.

Contextual Importance and Utility (CIU) were proposed by Kary Främling in 1995 as concepts that would allow explaining recommendations or outcomes of decision support systems (DSS) to domain specialists as well as to non-specialists (Främling and Graillot 1995; Främling 1996). The real-world use case considered was to select a waste disposal site for ultimate industrial waste in the region of Rhône-Alpes in France, among the thousands of possible candidates. A first challenge to solve was how to build a good DSS that represents a good compromise between the preferences of all stakeholders. The next challenge was how the results of the DSS could be justified and explained to decision makers as well as inhabitants who were living close to the proposed site. Several DSS models with different underlying principles were developed and tested, notably weighted sum, Analytic Hierarchy Process (AHP), Electre I and a classical rule-based system. An approach using Neural Networks (NN) was also used, where the goal was to build the DSS model based on learning from data from existing sites, which then became a challenge of explaining black-box behaviour (Främling 2020a). The experience gained from that project allowed to identify the following requirements for the Explainable AI (XAI) method to develop:

1. It has to be *model-agnostic*.
2. It has to support different *levels of abstraction* because all detailed inputs used by the DSS do not make sense to all target explainees.
3. The *vocabulary* used for producing explanations has to be independent from the internal operation of the black-box because different users have different background knowledge, so the used vocabulary, visualisation or other ways of explanation should be adapted accordingly.
4. Inputs are typically not independent of each others in real-world cases, i.e. the value of one input can modify the *importance* of other inputs, as well as the *utility* of values of other inputs. Therefore, the method must be situation- or *context*-aware.

The Collins dictionary defines *importance* as follows: ‘The *importance* of something is its quality of being significant, valued, or necessary *in a particular situation*’. The *Context* in CIU corresponds to the ‘in a particular situation’ part of this definition. So, by definition, ‘importance’ is context-dependent. When dealing with black-box systems, the word ‘something’ signifies an input feature or combination of input features. Therefore, CI could also have been called ‘Contextual Feature Importance’.

However, ‘importance’ does NOT express positive or negative judgements. Something like ‘good importance’, ‘bad importance’, ‘typical importance’, etc., does not exist. Adjectives such as ‘good’, ‘bad’, ‘typical’, ‘favorable’, etc., are used for expressing judgments about feature *values*, which leads us to the notion of *value utility* and *utility function*. The Collins dictionary defines a utility function as ‘a function relating specific goods and services in an economy to individual preferences’. In CIU, CU expresses to what extent the current feature value(s) contribute to a high output value, i.e what is the utility of the input value for achieving a high output value. Therefore, CU could also have been called ‘Contextual Value Utility’.

The next section provides the definition of CIU and implementation details. The following sections show results on classification and regression tasks with continuous- and discrete-valued inputs. Section 5 shows how to construct explanations using vocabularies and different levels of abstraction, followed by Conclusion.

## 1 Contextual Importance and Utility (CIU)

This presentation of CIU is based on the one in (Främling 2020b) and reuses the definitions found there, together with extensions and implementation details for the ‘ciu’ package. The ‘ciu’ package is available at https://github.

com/KaryFramling/ciu[1]. The most recent development version can be installed from there directly by the usual `devtools()` commands, as explained in the repository. Version 0.1.0, which is described here, is available from CRAN since November 20th, 2020, at https://cran.r-project.org/web/packages/ciu and can be installed as an ordinary R package using the `install_packages()` function. We begin with basic definitions for describing CIU.

**Definition 1** (Black-box model)**.** *A black-box model is a mathematical transformation $f$ that maps inputs $\overrightarrow{x}$ to outputs $\overrightarrow{y}$ according to $\overrightarrow{y} = f(\overrightarrow{x})$.*

**Definition 2** (Context)**.** *A Context $\overrightarrow{C}$ defines the input values $\overrightarrow{x}$ that describe the current situation or instance to be explained.*

**Definition 3** (Pre-defined output range)**.** *The value range $[absmin_j, absmax_j]$ that an output $y_j$ can take by definition.*

In classification tasks, the Pre-defined output range is typically $[0, 1]$. In regression tasks the minimum and maximum output values in a training set often provide a good estimate of $[absmin_j, absmax_j]$.

**Definition 4** (Set of studied inputs for CIU)**.** *The index set $\{i\}$ defines the indices of inputs $\overrightarrow{x}$ for which CIU is calculated.*

**Definition 5** (Estimated output range)**.** *$[Cmin_j(\overrightarrow{C}, \{i\}), Cmax_j(\overrightarrow{C}, \{i\})]$ is the range of values that an output $y_j$ can take in the Context $\overrightarrow{C}$ when modifying the values of inputs $x_{\{i\}}$.*

**Definition 6** (Contextual Importance)**.** *Contextual Importance $CI_j(\overrightarrow{C}, \{i\})$ expresses to what extent variations in one or several inputs $\{i\}$ affect the value of an output $j$ of a black-box model $f$, according to*

$$CI_j(\overrightarrow{C}, \{i\}) = \frac{Cmax_j(\overrightarrow{C}, \{i\}) - Cmin_j(\overrightarrow{C}, \{i\})}{absmax_j - absmin_j} \tag{1}$$

**Definition 7** (Contextual Utility)**.** *Contextual Utility $CU_j(\overrightarrow{C}, \{i\})$ expresses to what extent the current input values $\overrightarrow{C}$ are favorable for the output $y_j(\overrightarrow{C})$ of a black-box model, according to*

$$CU_j(\overrightarrow{C}, \{i\}) = \frac{y_j(\overrightarrow{C}) - Cmin_j(\overrightarrow{C}, \{i\})}{Cmax_j(\overrightarrow{C}, \{i\}) - Cmin_j(\overrightarrow{C}, \{i\})} \tag{2}$$

CU values are in the range $[0, 1]$ by definition. $CU = 0$ signifies that the current $x_{\{i\}}$ value(s) is the least favorable for the studied output and $CU = 1$ signifies that the $x_{\{i\}}$ value(s) is the most favorable for the studied output. In the CIU bar plots shown in the following sections, a 'neutral' CU of 0.5 corresponds to a yellow colour.

[1]A Python CIU package is available at https://github.com/TimKam/py-ciu and described in (Anjomshoae, Kampik, and Främling 2020). "R" and Python implementations are developed in the same team but independently of each other. The underlying CIU method is identical but the packages provide different kinds of visualisation and explanation mechanisms. Implementation details might also differ, such as how the 'Set of representative input vectors' is generated.

**Algorithm 1:** Set of representative input vectors

```
   Result: $N \times M$ matrix $S(\overrightarrow{C}, \{i\})$
1  begin
2      forall discrete inputs do
3          $D \leftarrow$ all possible value combinations for
             discrete inputs $\{i\}$;
4          Randomize row order in $D$;
5          if $D$ has more rows than $N$ then
6              Set $N$ to number of rows in $D$;
7          end
8      end
9      forall continuous-valued inputs do
10         Initialize $N \times M$ matrix $R$ with current input
             values $\overrightarrow{C}$;
11         $R \leftarrow$ two rows per continuous-valued inputs
             in $\{i\}$ where the current value is replaced by
             the values $min_{\{i\}}$ and $max_{\{i\}}$ respectively;
12         $R \leftarrow$ fill remaining rows to $N$ with random
             values from intervals $[min_{\{i\}}, max_{\{i\}}]$;
13     end
14     $S(\overrightarrow{C}, \{i\}) \leftarrow$ concatenation of $\overrightarrow{C}$, $D$ and $R$,
         where $D$ is repeated if needed to obtain $N$ rows;
15 end
```

**Definition 8** (Intermediate Concept)**.** *An Intermediate Concept names a given set of inputs $\{i\}$.*

CIU can be estimated for any set of inputs $\{i\}$. Intermediate concepts make it possible to specify vocabularies that can be used for producing explanations on any level of abstraction. In addition to using Intermediate Concepts for explaining $y_j(\overrightarrow{C})$ values, Intermediate Concept values can be further explained using more specific Intermediate Concepts or input features. The following defines *Generalized Contextual Importance* for explaining Intermediate Concepts.

**Definition 9** (Generalized Contextual Importance)**.**

$$CI_j(\overrightarrow{C}, \{i\}, \{I\}) = \frac{Cmax_j(\overrightarrow{C}, \{i\}) - Cmin_j(\overrightarrow{C}, \{i\})}{Cmax_j(\overrightarrow{C}, \{I\}) - Cmin_j(\overrightarrow{C}, \{I\})} \tag{3}$$

*where $\{I\}$ is the set of input indices that correspond to the Intermediate Concept that we want to explain and $\{i\} \in \{I\}$.*

Equation 3 is similar to Equation 1 when $\{I\}$ is the set of all inputs, i.e. the range $[absmin_j, absmax_j]$ has been replaced by the range $[Cmin_j(\overrightarrow{C}, \{I\}, Cmax_j(\overrightarrow{C}, \{I\}]$. Equation 2 for CU does not change by the introduction of Intermediate Concepts. In other words, Equation 3 allows the explanation of the outputs $y_j(\overrightarrow{C})$ as well as the explanation of any Intermediate Concept that leads to $y_j(\overrightarrow{C})$.

The range $[Cmin_j(\overrightarrow{C}, \{i\}), Cmax_j(\overrightarrow{C}, \{i\})]$ is the only part of CIU that cannot be calculated directly. A model-agnostic approach is to generate a *Set of representative input vectors*.

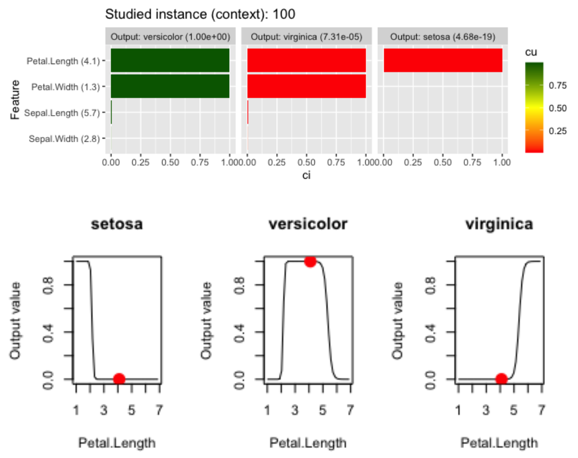

Figure 1: CIU explanation for instance #100 of Iris data set, including input/output plots for 'Petal Length'.

**Definition 10** (Set of representative input vectors)**.** *$S(\vec{C}, \{i\})$ is an $N \times M$ matrix, where $M$ is the length of $\vec{x}$ and $N$ is a parameter that gives the number of input vectors to generate for obtaining an adequate estimate of the range $[Cmin_j(\vec{C}, \{i\}), Cmax_j(\vec{C}, \{i\})]$.*

Algorithm 1 shows how the *Set of representative input vectors* is created in the current implementation. $N$ is the only adjustable parameter with a default value $N = 100$, which should be sufficient for most cases with only one input. The choice of $N$ is a compromise between calculation speed and desired accuracy.

If the input value intervals $[min_{\{i\}}, max_{\{i\}}]$ are not provided as parameters, then they are retrieved from the training data set by column-wise $min$ and $max$ operations. There is an obvious risk that this approach generates input value combinations that are impossible in reality. Whether that is a problem or not depends on how the black-box model behaves in such cases. In practice, this has not been a problem with the data sets studied so far, of which some are shown in the next sections. If such problems occur, then they can be dealt with in many different ways, however that remains out of the scope of this paper.

## 2 Classification with continuous-valued inputs

The Iris data set is used for this category mainly because the limits between the different Iris classes require highly non-linear models for correctly estimating the probability of the three classes for each studied instance. Figure 1 shows a CIU explanation generated for the 'lda' model and instance number 100 of Iris data set. The model was trained and figures were generated by the following code:

```
# No training/test set needed here
iris_train <- iris[, 1:4]
iris_lab <- iris$Species
model <- lda(iris_train, iris_lab)
ciu <- ciu.new(model, Species~., iris)
ciu$ggplot.col.ciu(iris[100,1:4])
```

The output values as a function of one input were generated by calling `ciu$plot.ciu()`. All other presented results have been generated in the same way.

The CIU explanation clearly shows that Petal Length&Width are the most important features separating 'versicolor' and 'virginica', where changing the value of either one would also change the classification. This is even the case for 'setosa', where a significantly smaller Petal Length value would change the result to 'setosa'.

## 3 Regression with continuous-valued inputs

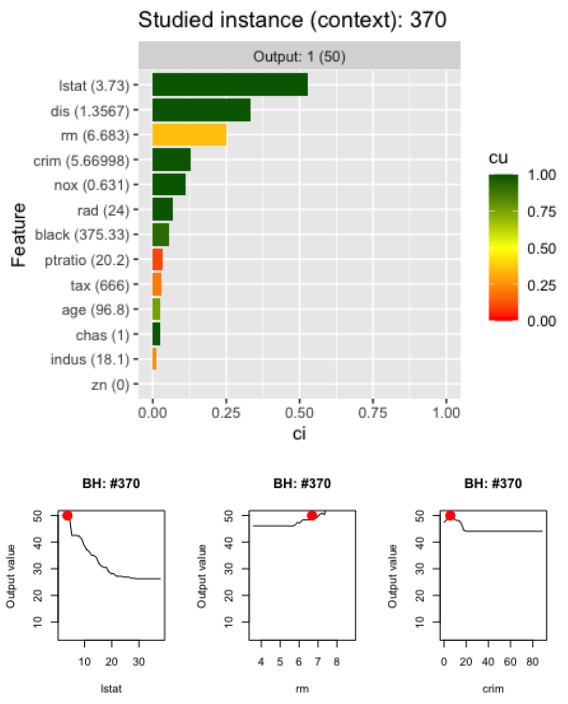

Figure 2: CIU explanation for instance #370 of Boston Housing data set, including input/output plots for features 'lstat', 'rm' and 'crim'.

The Boston Housing data provides a task of estimating the median value of owner-occupied homes in $1000's, based on continuous-valued inputs. It is a non-linear regression task that is frequently used for demonstrating results of XAI methods. Figure 2 shows CIU results using a Gradient Boosting Machine (gbm) model. The studied instance #370 is a very expensive one (50 k$) so the bar plot visualisation should be dominantly green, i.e. have favorable values at least for the most important features. This is indeed the case. However, there are also some exceptions and notably the number of rooms ('rm') is only average for this instance ($rm = 6.683$), even though such a value would be very favorable for most cheaper homes.

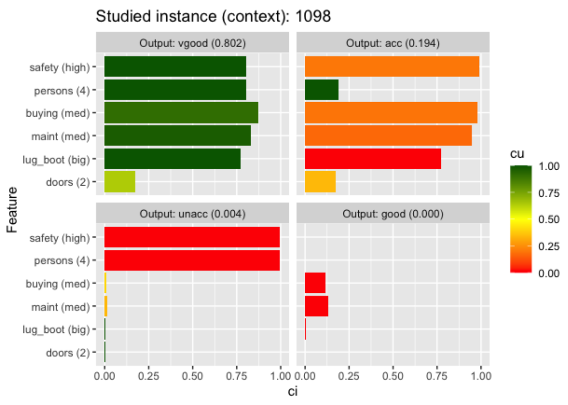

Figure 3: CIU explanation for Car instance #1098.

## 4 Discrete inputs

The UCI Cars Evaluation data set (https://archive.ics.uci.edu/ml/datasets/car+evaluation) evaluates how good different cars are based on six discrete-valued input features. There are four different output classes: 'unacc', 'acc', 'good' and 'vgood'. This signifies that both inputs and output are discrete-valued. Figure 3 shows the basic results for a 'vgood' car (instance #1098). The model is Random Forest.

CIU indicates that this car is 'vgood' because it has very good values for all important criteria. Having only two doors is less good but it is also a less important feature. In general, the CIU visualisation is well in line with the output value for all classes.

## 5 Abstraction Levels and Vocabularies

Using Intermediate Concepts, explanations can have different levels of detail and use different vocabularies. The initial results of the Cars data set were produced by the data set authors using a rule set that uses the intermediate concepts 'PRICE', 'COMFORT' and 'TECH', as reported in (Bohanec and Rajkovič 1988). The corresponding vocabulary is defined as follows in 'ciu'

```
price <- c(1,2)
comfort <- c(3,4,5)
tech <- c(comfort, 6)
car <- c(price, tech)
voc <- list("PRICE"=price,"COMFORT"=comfort,
        "TECH"=tech,"CAR"=car)
```

The vocabulary is provided as a parameter to the `ciu.new()` function. Explaining the output value using intermediate concepts PRICE and TECH is done by giving the parameter value `concepts.to.explain=c("PRICE", "TECH")` to the `ggplot.col.ciu()` method. If the explanation is for an intermediate concept rather than for the final result, then the 'target.concept' parameter is used as in

```
ciu$ggplot.col.ciu(cars.inst,
                   ind.inputs = voc$PRICE,
                   target.concept = "PRICE")
```

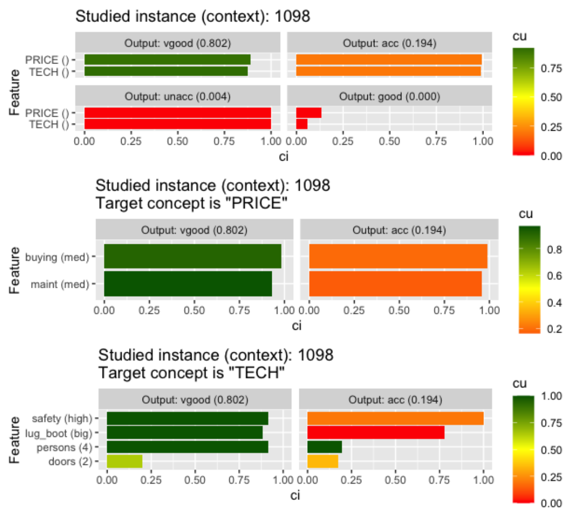

Figure 4: Car explanations using intermediate concepts.

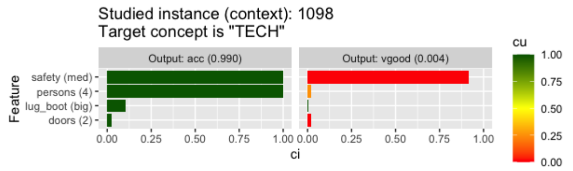

Figure 5: Car explanations after modifications.

The corresponding CIU explanations are shown in Figure 4. Figure 5 shows that the result changes from 'vgood' to 'acc' when changing the value of 'safety' to 'med' together with the corresponding TECH explanation. It is clear that if 'safety' is only 'med', then the car can't be 'vgood', no matter how good the other features are.

## 6 Conclusion

Contextual Importance and Utility make it possible to explain results of 'any' AI system without constructing an intermediate, interpretable model for explanation. Furthermore, CIU can provide explanations with any level of abstraction and using semantics that are independent of (or at least loosely-coupled with) the internal mechanisms of the AI system. The 'ciu' package is intended to allow researchers to apply CIU to all kinds of data sets and problems and to assess how useful the explanations are.

Work is ongoing on the use of CIU for image recognition and saliency maps and the results are promising. Those functionalities will be published in a new version of this package, or as separate packages.

## 7 Acknowledgments

The work is partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation..

## References

Anjomshoae, S.; Kampik, T.; and Främling, K. 2020. Py-CIU: A Python Library for Explaining Machine Learning Predictions Using Contextual Importance and Utility. In *Proceedings :*. URL https://sites.google.com/view/xai2020/home. Conference postponed from July 2020 to preliminary January 2021. .

Bohanec, M.; and Rajkovič, V. 1988. Knowledge Acquisition and Explanation for Multi-Attribute Decision. In *8th International Workshop on Expert Systems and Their Applications, Avignon, France*, 59–78.

Främling, K. 1996. *Modélisation et apprentissage des préférences par réseaux de neurones pour l'aide à la décision multicritère*. Phd thesis, INSA de Lyon. URL https://tel.archives-ouvertes.fr/tel-00825854.

Främling, K. 2020a. Decision Theory Meets Explainable AI. In Calvaresi, D.; Najjar, A.; Winikoff, M.; and Främling, K., eds., *Explainable, Transparent Autonomous Agents and Multi-Agent Systems*, 57–74. Cham: Springer International Publishing. ISBN 978-3-030-51924-7.

Främling, K. 2020b. Explainable AI without Interpretable Model. URL https://arxiv.org/abs/2009.13996.

Främling, K.; and Graillot, D. 1995. Extracting Explanations from Neural Networks. In *ICANN'95 Conference*. Paris, France. URL https://hal-emse.ccsd.cnrs.fr/emse-00857790.