0.0.1 Datasets

In this thesis, we use a multitude of datasets for generating the counterfactuals. All of the data sets where taken from Teinemaa et al. Each dataset consists of log data and contains labels which signify the outcome of a process. They were introduced by **[some author]**. The possible outcomes are *deviant* and *regular*. The first dataset is the popular BPIC12 dataset. This dataset was originally published for the Business Process Intelligence Conference and contains events for a loan application process. Each indivdual case relates to one loan application process and can be accepted (regular) or cancelled (deviant). In order to increase the speed of our experiments we limit the maximum sequence length to 25 events. We refer to this dataset as *BPIC12-25*. We also apply subsequent experiments on the same dataset with a maximum sequence length of 50**[To show validity regardless of sequence length]**. This dataset will be refered to as *BPIC12-50*. Lastly, we apply our approach to a third dataset of a different domain**[To show validity across datasets]**.**[Add a dataset description here.]** Below we list all the important descriptive statistics in Table 1. **[num deviant and num regular should be based on the counts within the cases.]** **[Time should just get seconds not this format.]**

dataset	BPIC12-25	BPIC12-50	BPIC12-Full
num cases	866	3728	4685
min seq len	15	15	15
max seq len	25	50	175
ratio distinct traces	0.001155	0.000268	0.000213
num distinct events	32	36	36
num data columns	23	25	25
num event features	21	23	23
time preprocess	0:00:01.327766	0:00:02.536075	0:00:03.541601
time unit	seconds	seconds	seconds
num deviant	14481	56996	99745
num regular	4305	64658	86948

Table 1: Table shows various statistics about the the datasets in use.

0.0.2 Representation

To process the data in subsequent processing steps, we have to discuss the way we will encode the data. There are a multitude of ways to represent a log. We will discuss 4 of them in this thesis.

First, we can choose to concentrate on *event-only-representation* and ignore feature attributes entirely. However, feature attributes hold significant amount of information. Especially in the context of using counterfactuals for explaining models as the path of a process instance might strongly depend on the event attributes. Similar holds for a *feature-only-representation*

The first is a *single-vector-representation* with this representation we can simply concatenate each individual representation of every original column. This results in a matrix with dimensions (case-index, max-sequence-length, feature-attributes). The advantage of having one vector is the simplicity with which it can be constructed and used for many common frameworks. Here, the entire log can be represented as one large matrix. However, eventhough, it is simple to construct, it is quite complicated to reconstruct the former values. It is possible to do so by keeping a dictionary which holds the mapping between original state and transformed state. However, that requires every subsequent procedure to be aware of this mapping.

Therefore, it is simpler to keep the original sequence structure of events as a seperate matrix and complementary to the remaining event attributes. If reqruired, we turn the label encoded activities ad-hoc to one-hot encoded vectors. Thus, this *hybrid-vector-representation* grants us greater flexibility. However, we now need to process two matrices. The first has the dimensions (case-index, max-sequence-length) and the latter (case-index, max-sequence-length, feature-attributes). **[This requires a change into formal symols that were defined prior.]**

0.0.3 Preprocessing

To prepare the data for our experiments, we employed basic tactics for preprocessing. First, we split the log into a training and a test set. The test set will act as our primary source for evaluating factuals, that are completely unknown to the model. We further split the training set into a training set and validation set. This procedure is a common tactic to employ model seletection techniques. In other words, Each dataset is split into 25% Test and 75 remaining and from the remaining we take 25 val and 75 train.

First, we filter out every case, whose' sequence length exceeds 25. We keep this maximum threshold for most of the experiments that forucs on the evolutionary algorithm. The reason is . Furthermore, two components of the proposed viability measure have a run time complexity of at least 2. Hence, limiting the sequence length saves a substantial amount of ressources.

Next, we extract time variables if they are not provided in the log in the first place. Then, we convert all binary columns to the values 1 and 0. **[In cases know time relevant information is avaible, we will XXX...]**

Each categorical variable is converted using binary encoding. Binary encoding is very similar to onehot encoding. However, it is still distinct. Binary encoding uses a binary representation for each class encoded. This representation saves a lot of space as binary encoded variables are less sparse, than one-hot encoded variables. **[from different from onehot encoding as we, encode each categorical as binary value rather than .]**

We also add an offset of 1 to binary and categorical columns to introduce a symbol which represents padding in the sequence. All numerical columns are standardized to have a zero mean and a standard deviation of 1.

We omit the case id, the case activity and label column from this preprocessing procedure, for reasons explained in subsection 0.0.2. The case activity are label-encoded. In other words, every category is assigned to a unique integer. The label column is binary as we focus on outcome prediction.

The entire pipeline is visualized in Figure 1.

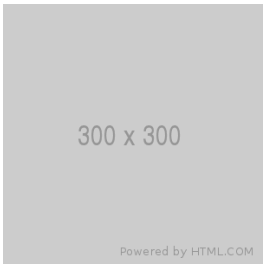

Figure 1: shows the entire preprocessing pipeline based on an example case.