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]**. We focus on binary outcome predictions. Hence, each dataset will provide information about one of two possible outcomes associated with the case. For instance, a medical process might be deemed a success if the patient is cured or a failure if the patient remains ill. A loan application process might deem granting the loan a success or the rejection as failure. The determination of the outcome depends on the use-case and the stakeholders involved. A insurence provider might deem a successful claim as a failure, while the client deems it as a success.

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 (success) or cancelled (deviant).

The next dataset is the Sepsis-Dataset.

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 **??**. **[num deviant and num regular should be based on the counts within the cases.]** **[Time should just get seconds not this format.]**

Dataset	Num Cases	Min Seq Len	Max Seq Len	Ratio Distinct Traces	Num Dist
Dice4EL	3728	15	50	0.000268	
BPIC12-25	866	15	25	0.001155	
BPIC12-50	3728	15	50	0.000268	
BPIC12-75	4461	15	75	0.000224	
BPIC12-100	4628	15	100	0.000216	
Sepsis25	707	5	25	0.001414	
Sepsis50	770	5	47	0.001299	
Sepsis75	777	5	66	0.001287	
Sepsis100	779	5	88	0.001284	
TrafficFines	129615	2	20	0.000008	

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.]**

0.3 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 section 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.

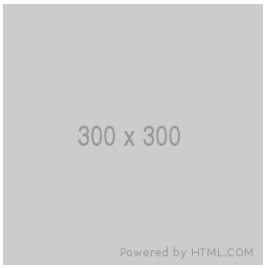

Figure 1: shows the entire preprocessing pipeline based on an example case.