|  |  |  |
| --- | --- | --- |
| **Fall 2024** | **Report #1 – 09/03/2024** | **Dingyi Nie** |

**——————————————————————————————————————**

**I. Task Achieved Last Week**

**——————————————————————————————————————**

* + Compared different imputing methods on P19 dataset.
  + Collected all potential dataset to be used on modeling time series with missing values.

**——————————————————————————————————————**

**II. Feedback and Interaction**

**——————————————————————————————————————**

* **Prof. Kuo’s Feedback**
  + How XGBoost handles missing values exactly

**——————————————————————————————————————**

**III. Report**

**——————————————————————————————————————**

**Imputer Comparison**

As I get back to work on this project after a while, I am able to examine previous approaches in a more objective and critical perspective. The first thing that comes to my mind is, how effective is our imputer?

To get rid of the missing values (NaN’s) in the input data, we have implemented multiple imputers:

* *Zero* imputer: replace all NaN’s with 0’s;
* *Mean* imputer: replace all NaN’s with the average observed value of the same feature on the whole dataset;
* *Forward* imputer: replace all NaN’s with the latest observed value of the same feature in the current time series; if no previous value is observed, replace it with the mean;
* *Linear* imputer: replace all NaN’s using the linear imputation of the 2 closest observed values of the same feature before and after; if only one value of the same feature is observed in the current time series, replace all NaN’s with that value; if no previous value is observed, replace it with the mean;
* *Spline* imputer: similar to Linear, but instead of linear imputation, do spline imputation;
* *Kmeans* imputer: imputation with this imputer is a 2-stage process. First, impute the whole dataset with spline imputation. Slice the dataset into same sized patches with a stride of 1. Sort the patches by “confidence” (proportion of NaN’s in the patch). Divide the sorted patches into several bins. Within each bin, do a K-means clustering, and calculate each cluster’s average value for each feature. Then, for imputing, replace NaN’s with conf \* + (1 – conf) \* , where is the spline imputation and is the cluster-mean.
* *XGBoost* imputer: slice the dataset into same sized, odd-length patches with a stride of 1. For each feature, select all patches that has an observed value at the center, and fit an XGBoost regressor that maps the input patch to the observed center value. Then use the regressor to impute all missing value.

Previously, we have learned that XGBoost as the classifier performs extremely well on P19 sepsis occurrence classification. Fit an XGBoost classifier to a dataset imputed by each of the imputers (including no imputation) above and we get this table:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Imputer | None | Zero | Mean | Forward | Linear | Spline | Kmeans | XGBoost |
| XGBoost AUROC | 95.26% | 94.69% | 94.60% | 93.78% | 93.18% | 92.86% | 94.18% | 91.51% |

Turns out that without any imputation, the XGBoost classifier will perform the best.

This implies that as suggested in [GRU-D](https://arxiv.org/abs/1606.01865), the missing pattern in P19 may actually be meaningful, i.e. there might be correlation between some features’ missing rates and the classification label distribution. According to the authors of GRU-D, this phenomenon occurs in PhysioNet Challenge 2012 and MIMIC-III, and it holds true to many other biomedical datasets. Hence, instead of designing a deliberate algorithm of imputation to best recover the fully observed data, we may want a higher-level structural design that enables exploiting the meaningfulness in the missing patterns with Green Learning modules.

**A List of All Datasets, Tasks and Specifications**

On P19 Sepsis occurrence prediction task, we observed that although green classifier performs better than most of the deep learning-based method, a simple XGBoost classifier still outperforms everyone else. We want to verify if the phenomenon also occurs on other similar datasets. However, since time series, specifically time series with missing values are an under-explored area, there’s few to none well-known benchmarks, and previous works on this topic typically do not share the same evaluation datasets and tasks. However, we do find a series of works from the deep learning community in recent years, that share some similarity in dataset characteristics and task options. Here’s an inclusive summary of them:

* [PhysioNet Challenge 2019 (P19)](https://physionet.org/content/challenge-2019/1.0.0/): Sepsis early prediction dataset. It is an hourly sampled dataset, so no resampling is needed. However, for metrics, the original P19 has a utility function for scoring the prediction based on temporal precision. But here we treat it as a simple binary classification task, and **the binary labels are assigned based on the occurrence of sepsis**, aligning with what's done in [RAINDROP](https://arxiv.org/abs/2110.05357) ([code](https://github.com/mims-harvard/Raindrop)). So theoretically we can compare our results so far with what's reported by RAINDROP, including their baselines;
* [PhysioNet Challenge 2012 (P12)](https://www.physionet.org/content/challenge-2012/1.0.0/): an irregularly sampled time series database. It has multiple outcome descriptors, but is mainly used for binary classification task such as in [GRU-D](https://arxiv.org/abs/1606.01865) ([code](https://github.com/PeterChe1990/GRU-D)), RAINDROP and [ViTST](https://arxiv.org/abs/2303.12799) ([code](https://github.com/Leezekun/ViTST)). The difference though, is that **GRU-D and ViTST both use mortality prediction as their primary task** (GRU-D also experimented on mortality + 3 other tasks, including los < 3d, as a multitask); while **RAINDROP uses length of stay < 3 day as the primary task**. (See [issue #14](https://github.com/Leezekun/ViTST/issues/14) under ViTST's repo). Note that, in the work of GRU-D, for the baseline models (RF for example), time series are resampled into an hourly basis (with forward-filling and different imputation strategies), while GRU-D itself, RAINDROP and ViTST are able to handle irregular time series so no resampling happens. GRU-D uses the subset A only for some reason.
* [MIMIC-III](https://physionet.org/content/mimiciii/1.4/): a massive biomedical dataset mainly collected from ICU stay data. It is also irregularly sampled. GRU-D resamples it every 2 hours and uses it for binary mortality prediction only based on first 48-hour data. They use 17 specific features selected in [this benchmark work](https://www.sciencedirect.com/science/article/pii/S1532046418300716) ([code](https://github.com/USC-Melady/Benchmarking_DL_MIMICIII/tree/dep_notebooks)).

**——————————————————————————————————————**

**IV. Next Steps**

——————————————————————————————————————

* Explore correlation between each feature’s missing rate and sample label;
* P12 dataset is relatively straightforward to handle; repeat all previous experiments on it;
* Explore possible solution for irregularly sampled time series input.

**——————————————————————————————————————**

**V. Milestone**

——————————————————————————————————————

* Made a list of datasets and related works (mainly in deep learning), listed commonly used benchmark tasks and marked their specifications;
* Codes are available here: <https://github.com/d9sus4/GL-TS>