Challenge CFM

LACOMBE Yoach
MVA | Télécom Paris | HEC Paris

Abstract

Missing data imputation is a problem attracting more and more consideration from the scientific community. In this challenge, we were asked to predict missing daily spread from more than 300 futures contracts collected over a 10-year period, using spatio-temporal features such as trade volume or open-interest. Several imputation paradigms were used ranging from matrix completion paradigm to multiple imputation paradigm.

1. Introduction

We are interested in predicting missing daily spread using historical data from more than 300 futures contracts. Futures contracts are contracts determining the delivery of an underlying asset at a future time for a predetermined price. The underlying asset can be virtually any asset. The bid-ask spread is the difference between the highest price a buyer is ready to pay for the futures contract (highest bid price) and the lowest price a seller will accept (lowest ask price) for a given asset. When the bid-ask spread is zero, supply meets demand and a trade is executed.

The daily spread is the average of the spread throughout the day. It is a good indicator of the liquidity of the market, as a large bid-ask spread typically means fewer buyers and sellers and a small bid-ask spread means people are ready to trade. Note that the bid-ask spread is defined as the number of ticks separating buying and selling price, where a tick is the minimum upward or downward movement in the price of the future (i.e it is the unity of the futures price). However, the daily spread is not an integer as it is the mean spread of the day.

1.1. Dataset description

We were given a dataset with a little less than 1.000.000 rows, and around 20 percent of missing spread. For each futures contract (uniquely identified by the underlying and the liquidity rank - i.e the rank from the closest contract to expire to the latest contract to expire), we have:

- the corresponding day number (*dt_close*).
- the date of the expiry (*dt_expiry*).
- the contracts prices at market opening, market closing, the highest and lowest prices during the day (resp *open*, *close*, *high*, *low*).
- the number of contracts exchanged during the day (*volume*).
- the number of active contracts at the end of the day (*open_interest*).
- the tick size (*tick_size*) as well as booleans indicating if the trading day is normal and if one or more features have been fixed (resp. *normal_trading_day*, *fixed*).
- the *spread* (which can be missing or not).

1.2. Initial observations

While we seem to have a lot of data, i.e 10-year historical data from 365 different futures contracts making up for 859.915 entries, it turns out that we have only from 226 to 2539 observations per contract, with around 26.9% of missing value per contracts.

Thus the main difficulty here will be to draw from both temporal (past and future known spreads and high, low, open, close prices) and spatial (other contracts current spreads and current high, low, open, close prices) information to predict the missing spreads.

To confirm this observation, Figure 1 presents the correlation between the per-contracts spreads and the other features and Figure 2 presents the auto-correlation of the per-contracts spreads at different lags. We can here observe both a temporal and spatial correlation, thus indicating that the best performing imputation algorithms should draw information from spatio-temporal features.

Also note that by simply plotting some contracts spreads, one can note that some spreads seem stable with few variations, other seem periodic and some seem to have no pattern. Lastly, for every instruments there are days with out-of-range spreads, i.e days with extraordinary behavior.

1.3. Literature review

I identified the problem as a multivariate time-series missing data imputation. This problem is essential for

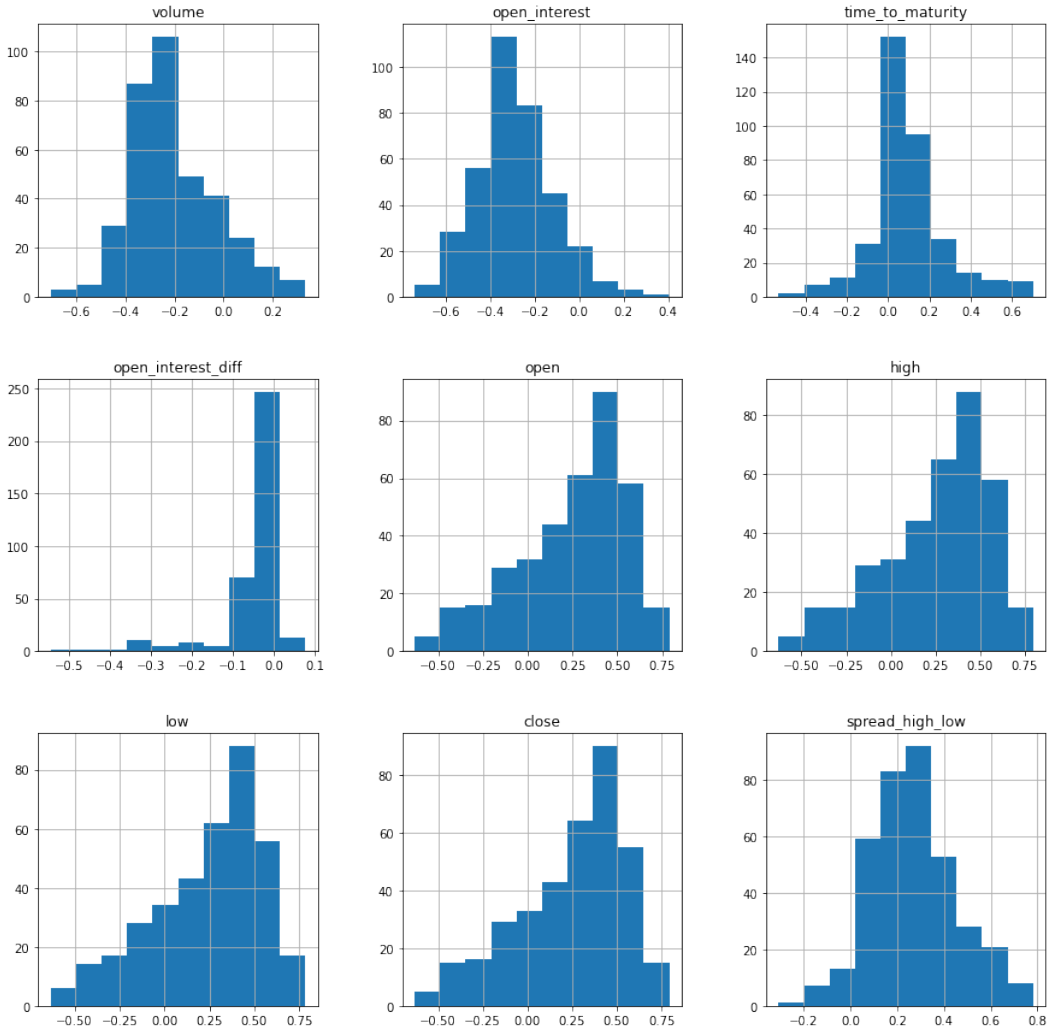

Figure 1. Histogram of the per-instrument correlations between the spread and other features.

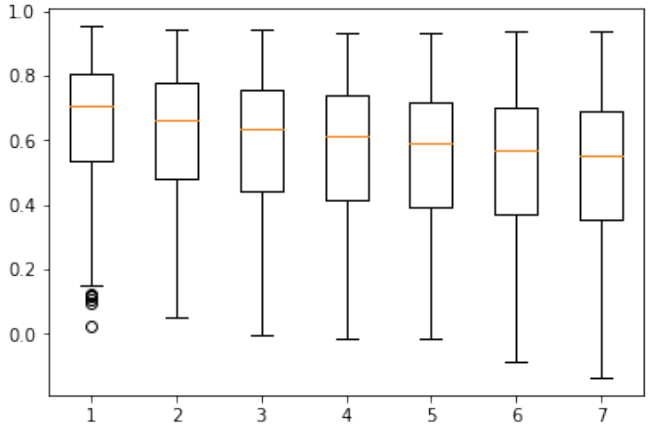

Figure 2. Boxplot of the per-instrument auto-correlations of the spreads at different lags.

health studies as clinical time-series often suffer from sparsity, irregularity, noise and missing data. Therefore, few studies have been led on financial time-series.

The paper [7] gives a good overview of already existing approaches. Namely, but not exhaustively, methods can be coarsely grouped as follows:

- **Deletion methods** - simply erasing missing data samples.
- **Neighbor based methods** - Imputes value from neighbors (i.e from the n closest samples) by taking the mean value of these neighbors. The neighbors are identified via clustering. The scikit-learn's KNNImputer [12] is a good example of such method.
- **Constraint based methods** - As the name indicates, these methods first establish some rules, before imputing the missing values while respecting those rules and constraints, for example, see [**?**]. Such methods work well with highly continuous data or clear patterns. It also seems that it should work well with categorical data or strings with clear rules (such as address).
- **Interpolation methods** - as the name indicates, interpolation (such as linear, cubic, quadratic, spline interpolations) is a possible method. Extremely simple, it is already a good benchmark. However, these methods are univariate, and so don't rely on inter-contracts (spatial) or intra-contracts correlations.
- **Regression based methods** - Classic autoregressive models such as ARX or ARIMA can be used to impute missing data. Some more advanced methods, such as LOESS or IMR (see [7] for the references) use regression model from nearest neighbors as well. However, these models rely on historical data, and do not capture information from the future data.
- **Statistical based methods** - We can either use simple statistics to impute missing data, such as the mean value or the median value, as well as more sophisticate statistic models (see [7] for the references). As [7] rightfully remarks, these methods rely on past and future information, thus capturing more information from the data than regression methods.
- **Matrix Factorization (MF) based methods** - MF based approaches decompose the data matrix into 2 low-dimensional matrices, allowing to extract features from the original data. By reconstructing the original matrix, we can impute missing values. While this approach doesn't rely on temporal correlation, some approaches add regularization to account for temporality (see the TRMF algorithm [15] or this SVD approach [8] for example). The Python *transdim* implements such approaches for transportation data.
- **Expectation-Maximization (EM) methods** - From a predefined model, the EM based methods (ex: [14]) consist in two steps - the Expectation step and the Maximization step, in order to find the parameters of the model the most suited to the present data. Then, these model fill the missing data with the most probable entry according to the model.
- **Deep learning (DL) methods** - There are a lot of DL-based approaches for missing data imputation. While the most basic ones consist only in MLP (multi-layer perceptron) or RNN [4] (and its derivate : GRU, LSTM etc), which can capture time information, the most sophisticate approaches goes from GAN (generative adversarial model), to graph-based approaches or Attention-based approaches. See [9] for a GAN-based method and a bidirectionnal-RNN method. See [2] or [13] for a transformer-based approach.

While imputation is a task not that well studied, time-series forecasting is thoroughly studied in the literature and some insights can be drawn from some of the results. Note that imputation can be seen as a bi-directional forecasting, as we are aware of both the past and the future, as well as some present context. My first thought when the challenge was presented to me was to use temporal fusion transformers (TFT [11]), a state-of-the-art approach for forecasting that incorporates static covariates, known futures inputs ad exogeneous time series. However, adapting such code and method for missing data imputation is a challenging task that I didn't have the time to address.

An interesting and recent result [6] from the forecasting literature argues that well-crafted gradient-boosting models can match and sometimes outperform the most sophisticated DL models on most of the tasks. Note that the TFT approach still outperforms every other forecasting algorithms.

2. Presentation of the algorithms I used

As pointed previously, the number of non-missing observations per contracts is quite low. In the following, I present the main algorithms I tried during the span of my studies (starting from March 1st to March 21th).

While the most efficient ones were relatively "simple" methods, I believe that, with a little more time and better computation power, DL-based approach could have outperform every other method.

Please also note that, while I tried many methods, a thorough investigation on one particular method, with proper parameters and features-selection tuning, could have yield better performances.

2.1. PPCA

Probabilistic PCA [14] is a modified approach of the PCA, a dimensionality-reduction algorithm that aims at lin-

early projecting the data on a lower dimensional latent space in which the first dimensions reduces the most the variance of the data.

PPCA assumes that each latent variable is normally distributed and uses a EM-algorithm to iteratively find the most suitable parameters.

I used the *pca-magic* Python library.

While this approach is fast and reasonably efficient, it suffers from a big inconvenience, i.e each time-stamp sample is suppose to be independent from the other.

The model

At each time stamp t_i, we consider a F-dimensional data point x_{t_i} where $F = 365\mathbf{x}k$. The first 365 coordinates here are the spreads of the 365 contracts at t_i, while the $(k-1)\mathbf{x}365$ next coordinates can be any set of features created from the other features of the original dataset - for example, the opening and closing prices, the time to maturity (i.e *dt_expiry - dt_close*).

We thus have a dataset $\mathbf{X} = \{x_{t_i}\}_i$ of T data points. We want to represent each x as a latent variable z with lower dimension $K \leq F$. We assume that $z \sim N(\mathbf{0}, \mathbf{I})$ and that x is generated via a projection, $x|z \sim N(\mathbf{W}\mathbf{z}_n, \sigma^2\mathbf{I})$, where $\mathbf{W}$ are the principal axes.

The rest of the algorithm applies the EM-method to estimate $\mathbf{W}$, with σ assumed to be known.

2.2. Modified MICE

MICE (Multiple Imputation by Chained Equations) [1], also called "sequential regression multiple imputation" is a flexible procedure in which a series of regression models are trained on each feature depending on the other feature of the dataset.

2.2.1 The algorithm

$\mathbf{X} \in \mathbb{R}^{T\mathbf{x}F}$ is a matrix containing some missing values. $\mathbf{X}[:, i]$ denotes the i-th feature's vector while $\mathbf{X}[:, -i] \in \mathbb{R}^{T\mathbf{x}(F-1)}$ denotes the $F-1$ other features' matrix.

1. Keep the mask matrix $\mathbf{M} \in \mathbb{R}^{T\mathbf{x}F}$ in memory, where $\mathbf{M}_{i,j} = 1$ if $\mathbf{X}_{i,j}$ is not missing and 0 otherwise.
2. impute the missing values of $\mathbf{X}$ with a simple procedure (either mean, median or linear interpolation).
3. chose a feature i that has missing values either at random or following a procedure.
4. take the indices $I_{present}$ where $\mathbf{M}[:, i] = 1$.
5. run a regression algorithm, with $\mathbf{X}[I_{present}, i]$ as the target variable and $\mathbf{X}[I_{present}, -i]$ as the training set.
6. predict $\mathbf{X}[I_{absent}, i]$ from $\mathbf{X}[I_{absent}, -i]$ and impute the values to $\mathbf{X}$.
7. repeat steps 3 to 6 for a number of cycles.

2.3. Some comments

MICE is an efficient and simple algorithm which fits iteratively a regression model on each feature while inducing some bias by imputing the missing values of the training data from either a simple imputation procedure (I've chosen linear interpolation) or from a previously fitted regression model.

I've chosen a gradient-boosting regression (XGBRegressor) algorithm as the regression model as it can dynamically chose the right features to predict the current target. Indeed, as previously pointed, some contracts benefits from a high-correlation with other futures spreads while other contracts spread can be more periodic (depending on the next maturity) and other contracts spread can be deduced simply from the volume/open interest variations. A gradient-boosting algorithm can simply learn from the best features according to each case, as long as the features were carefully engineered.

2.3.1 Features engineering

Here, I wanted to account for temporal and spatial dependency. From the initial dataset, I created the matrix $\mathbf{X}$ as explained in the model part of the PPCA section. I.e, I refactored the inital dataset as a univariate time-series for each tuple $(t_i, contract_j, feature_k)$ and created a dataset where each row corresponds to a time t_i and each column to the tuple $(contract_j, feature_k)$.

Because spreads can be auto-correlated, I wanted to account for short-term time dependency, so at step 3, I added each time a few columns corresponding to the spreads from time t_{i-k} to time t_{i+k}. However, these improvements did not yield better results. My hypothesis is that the univariate time dependency can already be deduced from the other features (the other spreads at current time, or the other features related to the current contracts).

2.4. GRIN

Deep-Learning approaches proved to be strong imputators [7]. However, most of the approaches of the litterature are either univariate or are not suited for time-series because samples are considered independent. To account for temporal correlation, RNN layers and other variants (GRU, LSTM, MPGRU) can be used. To account for spatial correlation is a much harder task, especially when the number of variables is large.

Several approaches can be used for spatial correlation. For example, deep-MVI [2] uses a kernel based method to add the closest other time-series when predicting the missing value of a given time-series.

Using a Graph-based approach could be more natural. Indeed, the structure of the market could fit a graph, where close contracts are close in the graph. GRIN (Graph Recurrent Imputation Network)[5] dwells on this approach by

proposing a message-passing multi-variate modeling of the imputation task.

More formally, one can define from the spread correlation between instruments a graph structure - for example, by creating the similarity matrix of the graph with a k-nearest graph. Once it is defined, one can pass the sequencs of graph to the GRIN model.

Unidirectional GRIN is composed of a spatio-temporal encoder (a message passing GRU) followed by a spatial decoding (a message-passing layer). Bi-directional GRIN 3 is simply composed of two unidirectional GRIN.

Modelisation: I first created a similarity matrix from a k-nearest graph given by the spread correlations between the futures contracts. I then created as usual $\mathbf{X} \in \mathbb{R}^{T \mathbf{x} F}$ and $\mathbf{M} \in \mathbb{R}^{T \mathbf{x} F}$, the time-series data matrix and the mask matrix. However, this time I only used spread (and no other features), so F = 365.

While this approach seems really promising and yields good results, I was not able to account for multiple features (for example volume, open interest and other prices) because of a lack of time - indeed adapting the code for this particular improvement is more difficult than expected. The authors seemed to believe that the approach would yield good results for multi-dimensional nodes, but have not implemented or tried such experiments.

2.5. Other approaches

I studied and implemented a numerous number of other approaches. The ones I tried are:

- **Linear interpolation** - It is the benchmark and already yields good results.

- **Deep-MVI** - Already mentioned before [2], Deep-MVI uses Transformers for time-specific tasks and kernel regression for spatial attention. Here again, I only used spread, as the algorithm is not suited for multi-dimensional nodes. The results I found were particularly bad, but with careful parameter search and features tuning, I believe it would have been a good approach.

- **TRMF** - It [15] uses an autoregressive (AR) temporal regularized matrix factorization. See the particularly good *transdim* library for similar approaches and clear code.

- **KNN-Imputer** - already mentionned earlier, KNN-Imputer is a simple, yet extremely efficient approach. For this particular task, I used the inital dataset almost as it was. I only added crafted features such as TTM (time to maturity), the difference between higher and lower daily prices. I also normalized some features.

Other approaches I started to work on but have abandoned are Temporal Fusion Transformers [11](that I wanted to adapt for Imputation), CDRec [10] or BRITS [3].

3. Results

See table 1 for the best results per method on the public test set. Note that my best method (i.e MICE) yields a final score of 0.6546, thus indicating that this method does not overfit the public set.

4. Interpretation and comments

My best performing algorithms are MICE, KNN-Imputer and PPCA. Note that each one of this algorithms rely on spatial correlation. The success of such methods suggest that the spatial correlation in this task is more important than the temporal correlation for missing spread imputation.

Indeed, as some plotting of a few spread curves led me to believe, the most difficult cases to impute are cases of daily market excitation or inhibitions, i.e days in which the market activity was exceptional. These particular days cannot be predicted from past or future spreads since they are by definition unrelated to previous or futures days. However, the same-day spreads of correlated instruments or the other features related to the current instruments (such as market closing, opening prices, or the volume, the open interest etc.) are good indicator of such exceptional activity.

That's also why I believe that with a little more work, GRIN could be a good fit for such a task, if only the other features were correctly incorporated to the model.

Ablation study: To quickly establish the truth of this hypothesis, I trained the MICE algorithm with *only* the spreads. In other words, with F = 365. I saw a drop of performances, going from 0.652 to 0.731 on the public test set RMSE. It confirms the importance of spatial correlation, especially the other features (OI, volume, closing, opening, highest and lowest prices, TTM etc.).

5. Conclusion

To conclude, I have studied and worked on a few missing data algorithms throughout the few weeks of my work. I have demonstrated that a simple algorithm such as MICE, coupled with a strong regressor such as XGBRegressor, and carefully thought feature engineering can yield really satisfactory results.

The task of missing data imputation is an interesting task that deserves more attention from the litterature, as it is closely related to the forecasting task, and as it teaches a lot on the nature of time-series.

My next line of work would be either to adapt the TFT [11] algorithm to fit an imputation task, or to adapt the GRIN algorithm by incorporating daily context (volume,

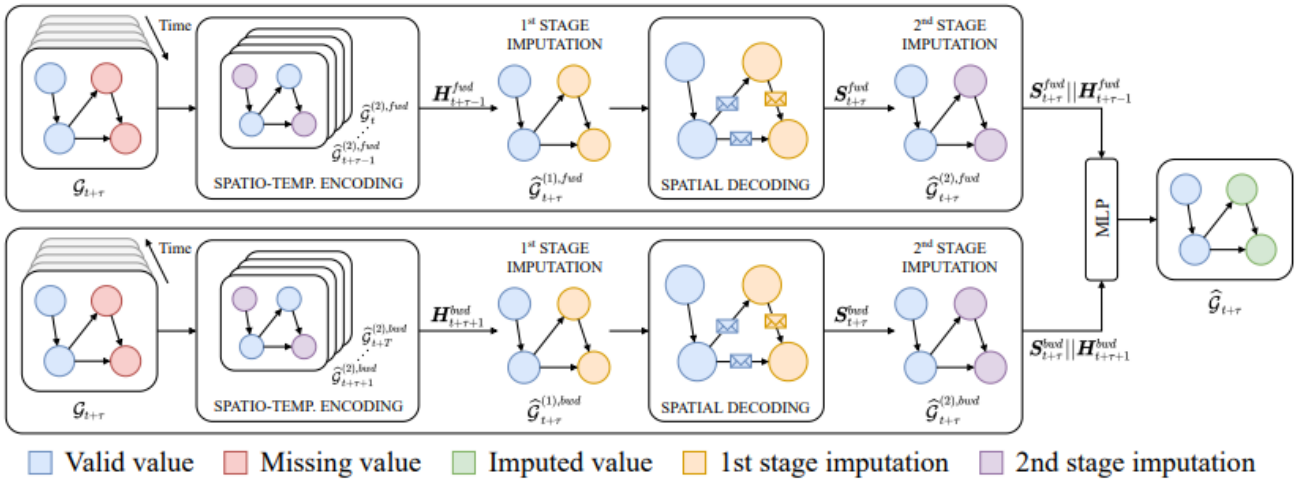

Figure 3. Overview of GRIN's architecture.

OI, open/close prices etc.).

References

[1] Melissa Azur, Elizabeth Stuart, Constantine Frangakis, and Philip Leaf. Multiple imputation by chained equations: What is it and how does it work? *International journal of methods in psychiatric research*, 20:40–9, 03 2011. 4

[2] Parikshit Bansal, Prathamesh Deshpande, and Sunita Sarawagi. Missing value imputation on multidimensional time series. *CoRR*, abs/2103.01600, 2021. 3, 4, 5

[3] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. Brits: Bidirectional recurrent imputation for time series. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 5

[4] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, 8, 04 2018. 3

[5] Andrea Cini, Ivan Marisca, and Cesare Alippi. Multivariate time series imputation by graph neural networks. *CoRR*, abs/2108.00298, 2021. 4

[6] Shereen Elsayed, Daniela Thyssens, Ahmed Rashed, Lars Schmidt-Thieme, and Hadi Samer Jomaa. Do we really need deep learning models for time series forecasting? *CoRR*, abs/2101.02118, 2021. 3

[7] Chenguang Fang and Chen Wang. Time series data imputation: A survey on deep learning approaches. *CoRR*, abs/2011.11347, 2020. 3, 4

[8] Trevor Hastie, Rahul Mazumder, Jason D. Lee, and Reza Zadeh. Matrix completion and low-rank svd via fast alternating least squares. *J. Mach. Learn. Res.*, 16(1):3367–3402, jan 2015. 3

[9] Daniel Jarrett, Jinsung Yoon, Ioana Bica, Zhaozhi Qian, Ari Ercole, and Mihaela van der Schaar. Clairvoyance: A pipeline toolkit for medical time series. In *International Conference on Learning Representations*, 2021. 3

[10] Mourad Khayati, Michael Böhlen, and Johann Gamper. Memory-efficient centroid decomposition for long time series. In *2014 IEEE 30th International Conference on Data Engineering*, pages 100–111, 2014. 5

[11] Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021. 3, 5

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 3

[13] Sindhu Tipirneni and Chandan K. Reddy. Self-supervised transformer for multivariate clinical time-series with missing values. *CoRR*, abs/2107.14293, 2021. 3

[14] Michael E. Tipping and Chris M. Bishop. Probabilistic principal component analysis. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 61(3):611–622, 1999. 3

[15] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. Temporal regularized matrix factorization for high-dimensional time series prediction. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. 3, 5

Method	Parameters	Public score
MICE	2 iter,200 trees,depth 3	0.651
KNN-Imputer	5 neighbors	0.689
PPCA	K=100, tol=1e-4	0.720
GRIN	with no global attention	0.723
Linear Interpolation	**BENCHMARK**	**0.891**
XGBRegressor	Naive approach	0.953
TRMF	K=30, 500 iter	1.12

Table 1. Best results on the public score of a few imputation algorithms.