Security & Privacy: Federated Summary algorithm

Authors: H. Alradhi, F.C. Martin, B. v. Beusekom

Introduction

This document is intended to assess the risk of using the Federated Summary algorithm. The document is modelled after the guidelines for describing risks for a federated learning algorithm as described in the vantage6 Security & Privacy document **[1]**.

The first section explains how the algorithm works and which data is shared between the parties. Note that we only discuss data that originates from the privacy sensitive database and not the data that is used by the vantage6 infrastructure. In the second part, we discuss which known federated learning vulnerabilities apply to this algorithm. Finally, we discuss how these vulnerabilities may be mitigated.

The algorithm

The summary algorithm is just that, it returns a summary of some dataset(s) much like what is present in R. Let D represent the culmination of dataset(s) where d_i is a column of D. Then the summary algorithm will return the following:

- Mean of d_i
- Standard deviation of d_i
- Number of missing rows of d_i
- Number of useable rows in d_i
- Range of d_i
    - If d_i is categorical or factor data this will return a table of counts per unique category
    - If d_i is numeric it returns the minima and maxima of the data

There are four types of parties involved in the algorithm; (1) The aggregator, (2) the data stations, (3) the client and (4) the vantage6 server [1]. We also present a flow diagram which explains the different steps of the algorithm which are then explained in the remainder of this section. Note that the server is not displayed as it merely acts as a communication hub between data station, aggregator and researcher.

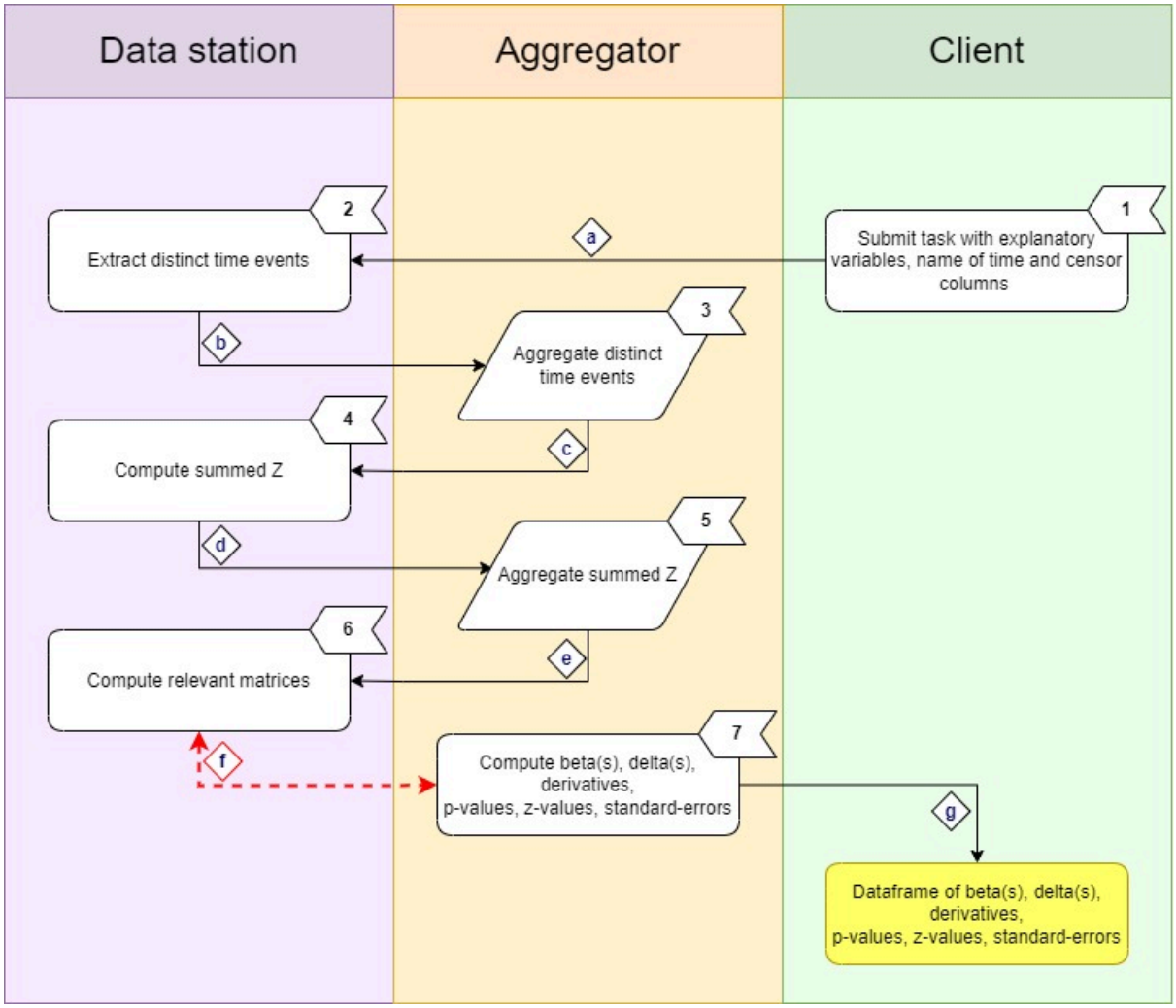

Figure 1 Schematic of the CoxPH algorithm. For a full description of the steps, see Section Algorithm Description. Note that all communication between these parties goes via Central Server.

Algorithm Description

The different steps of the algorithm are shown in Figure 1.

1. Once the researcher has established a suitable question and ensured that the column names match at each data station, they may initialise the algorithm.
2. The data stations extract the distinct event times and send their local contributions to the aggregator.
3. The distinct event times are aggregated by unique time so that there is a frequency table of the "global" distinct event times. The distinct times being sent are unique by appearance and not by individual record. For example, a data set with patients $[A, B, C, D, E]$ and corresponding time events $[1, 2, 1, 5, 1]$, the distinct times send to the aggregator from this data station are $[1,2,5]$. If we consider the case of two data stations, the first with the former set and the second with $[F, G, H, I, J, K]$ with event time $[1, 1, 3, 2, 7, 8]$, then the global distinct event times will be $[\text{"1"}: 3, \text{"2"}: 2, \text{"3"}: 1, \text{"5"}: 1, \text{"7"}: 1, \text{"8"}: 1]$.
4. Using information from step (1) the central server requests from the data stations to compute the summed Z statistic. For every explanatory variable included in the Cox Regression model, for the specific data station, this is just a sum of the specific variable with index label.
5. The results of (4) are sent to the aggregator where they are totalled, these will be referred to as global summed Z.
6. *NOTE: From this point on the algorithm enters a while loop till convergence or maximum iterations are reached, by default number of iterations is 30.* Once the aggregator computes global summed Z it enters a loop, it requests from each data station to compute three different matrices, namely:
    a. "agg1" which is the aggregate sum of the risk values for a specific event time.
    b. "agg2" which is the aggregate sum of the product of the prognostic index and risk values for a specific event time.
    c. "agg3" which is the aggregate sum of the product of pairs of the prognostic index and risk values for a specific event time.
    d. All of a-c are collated in a list and sent to the aggregator.
7. *NOTE: Algorithm still in loop.* The algorithm will compute estimates for the aggregated derivatives and following on from this, the model parameters, the Beta's, which are updated at each iteration and calculated in a stepwise manner using the primary and secondary derivatives. Then the algorithm computes the convergence criterion- if each successive

iteration's Beta(s) value is less than or equal to $delta = 10^{-8}$, stop otherwise iterate as it can conclude that the change in the primary derivative is not significant enough, thus this must be a good enough approximation for the beta, terminate. Else it continues, using the inverse of the negative of the secondary derivative it computes the Fisher Information Matrix (Hessian), the standard errors computed as the square-root of the Hessian, the z-value (Wald Statistic) which evaluates the significance of each parameter estimate, divide beta by its standard error to see how different the parameter is from 0. If there is a large standard error this will tend to 0 suggesting that the parameter is not very significant. Finally, it also calculates the p-value of each parameter.

Once converged or the iteration limit is met, the algorithm sends the results to the client in the form of a data frame containing the Beta(s), delta(s), standard error(s), z-value(s) and lastly the parameter p-value(s).

Data in transit

Table 1 – A description of data transfers between vantage6 components. Note that all data transfers are mediated by the vantage6 server. The risk level comes from the original paper on Security and Privacy ***[1]****.*

Description	Labels	Source	Destination	Risk
Initial input: explanatory variable names, names of time and censor columns	a	Client	Aggregator	Low
Distinct event times	b	Data station(s)	Aggregator	Low – High
Global distinct event times	c	Aggregator	Data station(s)	Low - High
Summed Z statistics	d	Data station(s)	Aggregator	Low
Aggregated summed Z statistics	e	Data station(s)	Aggregator	Low
Beta(s)	f	Data station(s)	Aggregator	Low
Primary derivative	f	Data station(s)	Aggregator	Low
Secondary derivative	f	Data station(s)	Aggregator	Low
Delta(s)	f	Data station(s)	Aggregator	Low
p-value(s)	f	Data station(s)	Aggregator	Low
z-value(s)	f	Data station(s)	Aggregator	Low
Standard error(s)	f	Data station(s)	Aggregator	Low
Data-frame containing the final version of all the 'f' label statistics	g	Aggregator	Client	Low

As is indicated in the table above, the transferred data types that are potentially most sensitive are the *distinct event times* per node as well as the global distinct event times as these are simply the aggregation of the node specific counterparts. However, whether these are sensitive in practice depends on how the analysis is executed.

For instance, if the client starting the analysis sends its personal unique time events, ($patients = [A, B, C, D, E], times = [1,2,3,4,5]$), the patient data is at risk because this allows the 1-to-1 linkage of patient to time.

All statistics that are derived from the distinct event times therefore have a potentially high risk. By using data that has multiple individuals per distinct event time, one-to-one linkage can be prevented. Then, the risk is low for all data types deriving from the event times. The risk for all other statistics is always low: they are derived from previously aggregated statistics, or they do not contain any data that can be traced back to the original data points. Most of these statistics are also computed in a non-federated scenario.

Risks

In this section we will look at the types of attack and other kind of risks that the algorithm will be vulnerable to. Not all types of attack are relevant to this algorithm. Please refer to the Security and Privacy document **[1]** for the various types of attack definitions. It is important to note that the risk analysis is not exhaustive and malicious parties will know more creative techniques.

Attack name	Risk analysis
Reconstruction	It is possible to reconstruct the distinct event times in case the attacker has access to the distinct event times of N-1 parties, where N is the total number of participating parties. It is impossible to reconstruct the patient variables from the numerator(s) and denominator(s).
Differencing	This potentially is possible through the preprocessing steps but not from this algorithm itself.
Deep Leakage from Gradients (DLG)	Not applicable.
Generative Adversarial Networks (GAN)	Not applicable.
Model Inversion	Not applicable.
Watermark attacks	Not applicable.

Mitigation

Considerations to reduce the impact in the event of a successful attack.

- Differential privacy: There are methods for defending against this such as adding Gaussian noise **[3]**. A suitable way to protect against this without the use of Gaussian noise would be to simply require at least three data stations for an analysis. By adding Gaussian noise to the data, it effectively adds a veil over the data ensuring that even if the attacker manages to reconstruct/differentiate, what they receive back is some perturbed data and not the original.
- Parties > 2: Doing this ensures that even if there is a malicious party, data differencing and reconstruction becomes extremely difficult as it is tricky to discern from the aggregated statistics which data belongs to each member of the collaboration.
- Minimal number of records per bin: To have some amount of control, a default of 3 records per distinct time event has been added. This is the default, but this can be increased by the data-owner. Doing this ensures further anonymity and confidentiality about the intermediary results of the analysis. **[4] [5]**
- Free selection of columns for the censor: There will be discrimination on the columns which can be censored. This means that column names at each individual data station must be correctly labelled.
- Enable the check that the distinct times are not equal to the number of records.

References

[1] Martin, F., van Gestel. A., van Swieten, M., Knoors D., van Beusekom, B., Geleijnse, G., 2023. 'Security and Privacy using vantage6 for Privacy Enhancing Analysis'.

[2] Lu, C.-L. et al. (2015) 'WebDISCO: A web service for distributed Cox Model Learning Without Patient-level data sharing', Journal of the American Medical Informatics Association, 22(6), pp. 1212–1219. doi:10.1093/jamia/ocv083.

[3] *Kairouz, P., Liu, Z. and Steinke, T., 2021, July. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In International Conference on Machine Learning (pp. 5201-5212). PMLR.*

[4] Data2Knowledge. (n.d.). Disclosure control. Retrieved, 2023, from https://data2knowledge.atlassian.net/wiki/spaces/DSDEV/pages/714768398/Disclosure+control, Authors: Butters, O., Wilson, B., Avraam, D., Westerberg, A., & Wheater, S.

[5] Richard, W,. 2019: SDC Handbook. Figshare. Book. https://doi.org/10.6084/m9.figshare.9958520.v1