Federated K-Means clustering

Swier Garst[1] and Marcel Reinders[1]

Delft University of Technology, 2600 AA, Delft s.j.f.garst@tudelft.com

Abstract. Federated learning is a technique that enables the use of distributed datasets for machine learning purposes without requiring data to be pooled, thereby better preserving privacy and ownership of the data. While supervised FL research has grown substantially over the last seven years, unsupervised FL methods remain scarce. This work introduces an algorithm which implements K-means clustering in a federated manner, addressing the challenges of data heterogeneity and convergence on harder datasets.

Keywords: Federated Learning · K-Means clustering · Distributed machine learning.

1 Introduction

Nowadays, lots of data is being generated in a distributed fashion. Mobile phones and other personal devices such as smart watches enable the collection of massive amounts of data. If made accessible, this data could prove useful for improving the performance of the services provided by these devices. However, due to a growing concern on data privacy, more and more users of these devices are hesitant in sharing their data. Furthermore, regulations such as the General Data Protection and Regulation (GDPR) act prevent the collection of data of this kind in bulk. Federated learning (FL) (McMahan et al. 2017) was introduced as a solution to this problem. In short, instead of pooling data to train a single model, instances of a model are being shared to data owners (clients), which then train the model on their local data. Then, these trained models are sent back to the central server, which aggregates the results. Next, a new round begins with the server sending out the updated models. This cycle continues until convergence.

Over the past couple of years, research has shown FL to be a promising technique, reaching performances comparable to a central approach in with all data of the clients is pooled at a single location) (Lee and Shin 2020 Sadilek et al. 2021). The vast majority of the federated learning research has been focusing on the supervised learning paradigm. Little work has been done on unsupervised federated learning methods, even less so when specifically looking into clustering techniques (T. Li et al. 2020 Q. Li et al. 2021). One of these clustering techniques is K-means clustering (Hartigan and Wong 1979). In a federated learning setting, K-means clustering can be described as trying to find overarching cluster

means over data which is distributed among different datasets (clients). Only a few papers discuss possible options for federated K-means clustering. Kumar, Karthik, and Nair 2020 introduce a method which assumes availability of some data at the central server to pretrain the model, which is not always feasible. Hou et al. 2021 use secure multiparty computation and blockchain to share encrypted data on which k-means is being performed. However, for some cases (e.g. in the medical domain), even sharing encrypted data might not be acceptable.

Bringing K-means into the federated domain comes with a couple specific challenges. First, not all clients need to have the same amount of clusters in their data. Therefore, they might also not have data from each cluster. Finally, due to this possible heterogeneity, a way of matching clusters from different clients is required. Liu et al. 2020 introduced a method for federated k-means in a setting where each client only holds one sample. Although this is highly valuable in the cross device setting, it does not address the challenges of varying amounts of local clusters. Servetnyk, Fung, and Han 2020 proposes a dual averaging approach for k-means clustering, which does not seem to address the challenge on how to match clusters from different clients. Dennis, T. Li, and Smith 2021 propose a one-shot method by doing a k-means clustering over the cluster means found locally. However, their experiments describe a static number of local clusters in the data available. Furthermore, the number of local clusters is an input parameter to their method, even though it is not always possible to determine this. Nevertheless, their choice to run a clustering centrally over cluster means from the local datasets inspired us to do this in a iterative approach. Next to that, we introduce a variable amount of local clusters. We show that both changes lead to improvements, especially in settings where the amount of local clusters is highly varying between clients. We also introduce a ruling which can automatically determine the amount of local clusters on each client, reducing the amount of parameters that need manual tuning. We show that in a 2-dimensional setting, a clustering of similar quality can be obtained compared to a conventional K-means clustering (on a pooled dataset).

2 Background

This work combines the K-means clustering algorithm and the federated learning paradigm. In this section, both these concepts will be discussed in more detail.

2.1 K-means clustering

The objective of a clustering algorithm is to partition a given dataset into several subsets with similar features. The K-means clustering algorithm does so by trying to minimize the so-called within cluster sum-of-squares criterion:

$$F_{km} = \sum_{j=0}^{m} \min_{\mu_k \in C} (||X_{j,k} - \mu_k||^2) \tag{1}$$

with m the amount of samples, μ_k the cluster mean of cluster k, C the set of all cluster means and $X_{j,k}$ being data point j assigned to cluster k. The procedure in which the k-means algorithm tries to minimize equation 1 consists of two steps. First, all data points get assigned to the cluster mean according to the lowest euclidean distance. Then, the mean value from all points assigned to a certain cluster is calculated. This is done for every cluster, creating a new set of means to start the next round with. This process is repeated until the change within these means is smaller than a certain threshold (and the algorithm has reached convergence) (Hartigan and Wong 1979).

2.2 Kmeans++

One of the drawbacks of classical K-means clustering is that its initialization is sampled uniformly from the underlying data. This means that having initial cluster means that all come from the same cluster is as probable as having initial cluster means spread across all clusters. Although the K-means algorithm itself can somewhat compensate for this, it still leads to large variability in performance. Arthur and Vassilvitskii 2006 developed an initialization method for K-Means to combat this high variability, called KMeans++. Instead of sampling K cluster means from the data with uniform probability, datapoints get weighted based on their distance to the closest already mean that is already chosen, with larger distances giving larger weights. This results in (on average) initializations that are more distributed over the space, and prevents (on average) initial cluster means from starting very close to each other, decreaseing K-means performance.

3 Methods

When federating the k-means clustering algorithm, clients perform k-means clustering on their local datasets, after which they send the resulting means to a central server. The server then aggregates these means in a certain way, before sending back a set of global means. Clients can then use these means as a starting point for their next iteration of k-means. However, compared with the supervised federated learning methods, several key challenges arise:

- **Cluster alignment** As the server receives the values for cluster means from different clients, there is not a set way of knowing which of these means can be assumed to be from the same global cluster. This is in contrast with averaging on a model such as a neural network, where due to the structure of the model, parameters can be averaged between models.
- **Imbalance in the numbers of clusters locally.** Datasets could be split in such a way that one client holds another set of clusters than another client. Not only is the number of clusters something that needs to be detected without looking at the data (as we cannot access said data from the central server), we also need to ensure that, when a client receives the entire set of new global means from the server, it only uses the ones corresponding to its' clusters.

Table 1. notations used

symbol	description
X^i	data on client i
K^g	global number of clusters
K^i	number of clusters on client i
C^g	global cluster means
C^i	cluster means of client i
S^i	amount of samples for each cluster on client i

Algorithm 1 The federated kmeans algorithm
Input: K_g

1: **Init:**
2: **on each client** $i \in N$ **do:**
3: $K_i = K_g$
4: C_i = kmeans++_init(X_i, K_i) ▷ get cluster means using kmeans++ initialization
5: send C_i to server
6: **For each round** r **do:**
7: **On server do:**
8: $C_l = [C_1|C_2|..|C_N]$ ▷ Concatenate all local cluster means
9: $S_l = [S_1|S_2|..|S_N]$ ▷ Repeat for sample amounts per cluster
10: C_g = kmeans(C_l, N_g, weights = S_l) ▷ Obtain new global clusters using kmeans
11: send C_g to all clients
12: **On each client** $i \in N$ **do:**
13: $C_i = C_g$
14: S_i = kmeans_assign(X_i,C_i) ▷ Determine empty clusters
15: $C_i = C_i$[s != 0 for s in S_i] ▷ Drop empty global clusters
16: S_i, C_i = kmeans(X_i, C_i) ▷ run kmeans using remaining global cluster means
17: send S_i, C_i to server

Here, we present a federated K-means algorithm that addresses these challenges. Notation used throughout this section is found in table 1. The pseudocode for the federated K-means algorithm can be found in algorithm 1. The algorithm can be divided into two parts: an initialization step, in which we generate initial cluster means on each client using kmeans++ initialization, and an iterative k-means step in which clients communicate their cluster means to the server, which aggregates these means into a 'global' set of means, which then gets redistributed to the clients for the next k-means iteration.

Determining the amount of local clusters.
While the global amount of clusters is set (main parameter k of the k-means procedure), it is not a given that each client has data for each of these clusters. In other words, the number of clusters between clients can differ, and is not necessarily equal to the number of clusters in the pooled data. In order to solve this problem, we determine which global clusters correspond to local data in each round on each client. Before a client makes a kmeans step, it assigns its data to the global cluster means it has received (line 14). Next, clients check if there

are empty clusters, i.e. cluster means which did not get any points assigned to them. If so, clients discard these empty clusters (line 15). The remaining (global) cluster means are then used as initialization for the next kmeans step (line 16). This way, k can locally become smaller when running kmeans on the clients.

Cluster alignment
After each client has calculated one iteration of K-means on their local data (each with their own amount of local clusters), they send their cluster means and amount of samples per cluster back to the server. The server then concatenates all cluster means, and aggregates them. It does so by running a K-means clustering on the received local means (using the global K parameter), to align clusters from different clients to each other. This global k-means step is weighted by the amount of samples per cluster found, such that a cluster with lots of samples in it will have a bigger impact on the aggregation step compared to a cluster with fewer samples. That is, we modify the objective function from equation 1 into:

$$F_{km} = \sum_{j=0}^{m} \min_{\mu_k \in C} (||s_k * X_{j,k} - \mu_k||^2) \tag{2}$$

where s_k is the amount of samples corresponding to cluster k. Note that the *cluster means* sent back by the clients are at the server used as the *samples* for clustering using k-means. Doing the aggregation with a k-means clustering, we solve the cluster alignment problem, since similar clusters will be close to each other and thus merged by the global k-means step.

Because the amount of samples have to be reported to the central server, there exists a privacy risk if a client finds a cluster with only one sample in it. To prevent this, any clusters holding only one sample are simply omitted from the list of means sent to the server.

4 Results

The federated approach with the local datasets (referred to as 'own method') is compared to the scenario where a K-means clustering is executed on all data centrally, as well as to the method of Dennis et. al., for which we tuned their parameter the local number of clusters (k_{local}) for each experiment. We also compare to their method using $k_{local} = k_{global}$, as it is not always possible to tune in the federated setting, as it would require looking at the data itself. We calculate the Adjusted Rand Index (ARI) for both central and federated approaches with respect to the labelled samples data. Since there are no labels for the clustering in the FEMNIST experiment, the silhouette score was used instead.

Clients holding different parts of the data.
In order to validate the federated K-Means algorithm, a synthetic two-dimensional dataset was generated. The generation procedure is taken from Servetnyk, Fung, and Han 2020. Sixteen cluster centers were chosen with an equal distance (here 5) from one another, see fig. 1(a). Then, 50 data points were sampled around each cluster center using a normal distribution (with variance 1). This data was then distributed among four clients in the following way: First, each client is assigned a 'location' within the field ($X_1, X_2 \in (-12.5, 12.5)$). From there, the probability P that a data point would be assigned to a certain client scales inversely with the euclidean distance d to that datapoint:

$$P = 1 - exp(-\frac{\beta}{d}) \tag{3}$$

where β is a parameter which can be tuned to promote more or less heterogeneity in the data separation. Differing from Servetnyk, Fung, and Han 2020, if a data point happens to be assigned to multiple clients, it instead gets assigned at random. Three different distributions of the data were generated, for $\beta = 0.1, 1, 10$. See figure 1 for the final distributions.
We generated three versions of this dataset. Note that β only changes which points get assigned to which client, meaning that it does not influence the performance for the central case. Figure 1(e) shows that our method is able to attain performance similar to a centralized kmeans clustering, while outperforming Dennis et. al., regardless of tuning. Performance seems to be independent of β, meaning that our algorithm is robust against more non-IID data distributions.

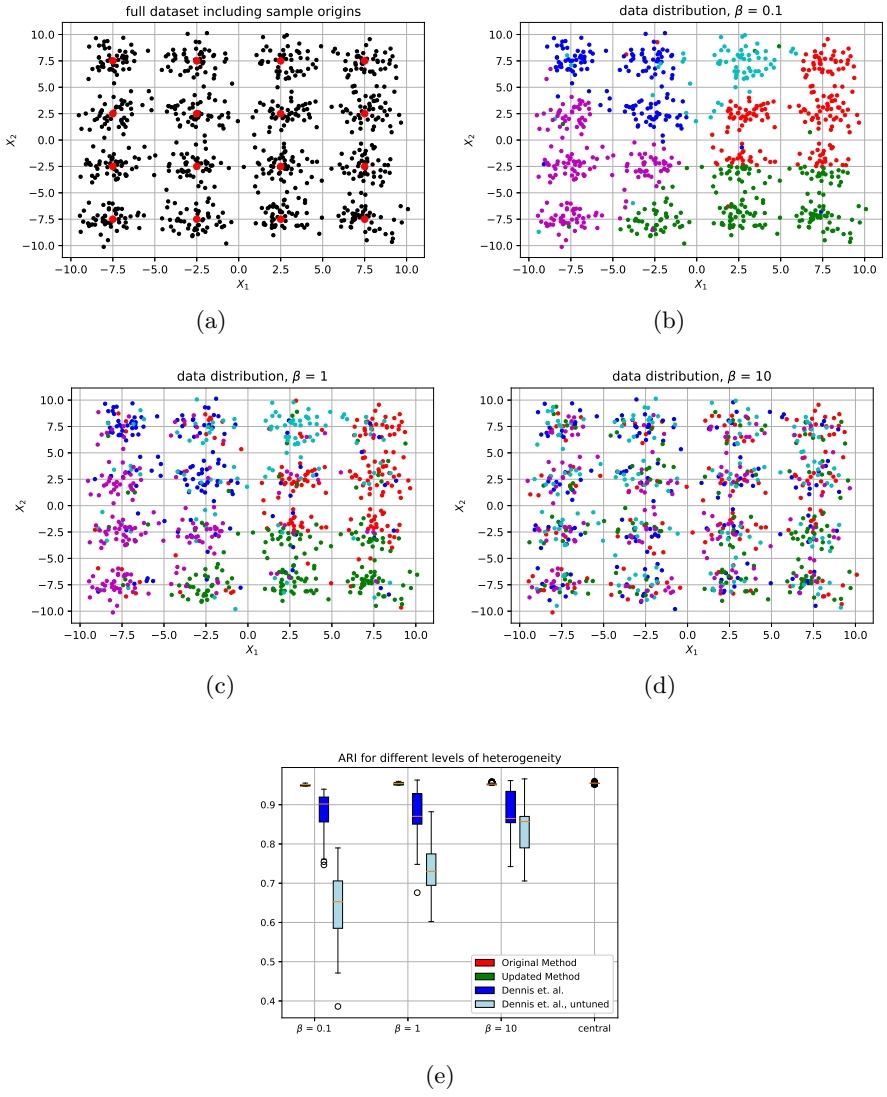


Fig. 1. The regular synthetic datasets. (a) shows the original sampling of the regular synthetic dataset, with the defined cluster means (from which the data are generated using a normal distribution N(0,1)) in red. (b) until (d) shows how this dataset is separated into five different subsets for different values of β. Different colors denote the different clients. (e) shows ARI results on all three datasets.

Increasing levels of noise.
Next, we explored the effect of having noisier clusters. We recreated the regular synthetic dataset, but varied the standard deviation from which samples are being generated, from 1 to 1.5 (original used 1). figure 2 shows the effect. We generated these datasets twice, once with 50 points per cluster and once with 200 points per cluster (original had 50).

Results on these datasets are shown in figure 3. For both central and federated clustering, the ARI scores go down for higher noise levels. This is expected, as there will be more points ending up closer to the cluster they did not originally belong to, meaning that even if kmeans finds the original cluster means perfectly, the label assignment will be off. Therefore, the relative difference between federated and central clustering is more important than the absolute ARI scores. Our method does seem to follow the same trajectory as a central kmeans clustering; however, variance seems to increase a lot compared to centralized clustering. Furthermore, for $\beta = 0.1$, mean ARI decreases compared to central clustering at high noise levels. The amount of points per cluster does not seem to influence ARI score significantly. Figure 4 shows that, especially for noisier datasets, there is a large benefit in being able to iterate more often.

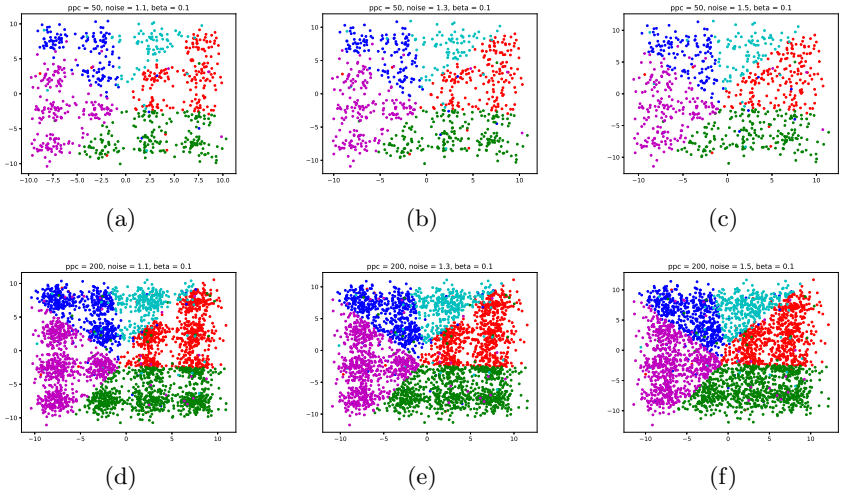


Fig. 2. Some of the distributions with increasing levels of noise, using 50 or 200 points per cluster.

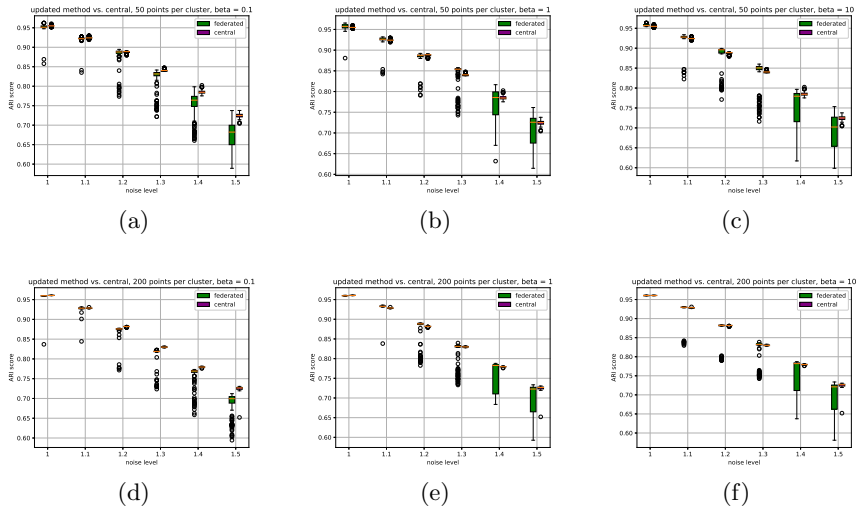


Fig. 3. results on using different levels of noise for different values of β, using 50 or 200 points per cluster.

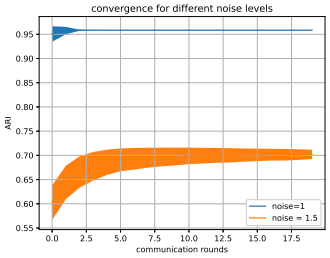
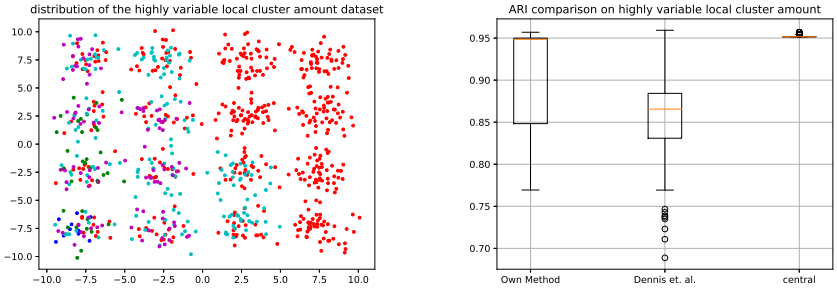


Fig. 4. ARI scores over time, showing a benefit to iterating for higher noise levels.

High variability in number of local clusters.
Next we wanted to explore the effect of having a variable local k. We used the same data as generated for the regular synthetic dataset, but distributed even more heterogeneously, such that each client only had data from 1, 4, 7, 10 or 16 clusters, respectively. See fig. 5(a).
Figure 5(b) shows that our method attains a similar mean as compared to the central case, however with a much larger deviation. This is probably caused by differences in initializations. If the algorithm initializes in such a way that clients assign data to more clusters than what is being present in their data, the algorithm has a hard time correcting for that. Furthermore, it does not help that one client only has ten datapoints in total, meaning it initializes ten clusters of size one, of which none are being send to the central server due to privacy issues.



(a) (b)

Fig. 5. Assessment of the method on data with a large variability of local clusters per client. (a) shows the distribution per client, (b) the ARI results for different methods.

Clustering higher dimensional real data.
In order to determine performance on a higher dimensional dataset, the Federated Extended MNIST (FEMNIST) from LEAF (Caldas et al. 2018) was used, which separates the original Extended MNIST (Cohen et al. 2017) handwritten numbers and letters based on the person who wrote them. Only 10 clients were used from the original FEMNIST, as this drastically sped up the experiments, while keeping enough data for a meaningful assessment. 6 shows that, outperforms Dennis et. al., regardless of tuning. There is still a difference with a central clustering, however.

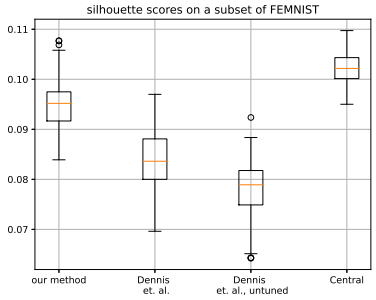


Fig. 6. silhouette scores on (a subset of) FEMNIST.

5 Discussion and conclusion

This work describes the implementation and validation of a federated K-means clustering algorithm, enabling clustering over multiple datasets without sharing the underlying data. Our results show performances approaching a central method, in which all data is brought into a single location. There are still some scenarios in which our method shows larger variability in performance as compared to a central clustering, however. These are mostly the more difficult scenarios, such as when there is an extreme distribution in the amount of cluster present on each client, or when the data has a high dimensionality as with the FEMNIST experiment. This could be improved by increasing the amount of iterations.

The FEMNIST experiments use the silhouette score as their performance metric. The silhouette score involves calculating distances from each point in a dataset to each other point in a dataset. This means that, to calculate a 'global' silhouette score, distances between datapoints from different clients need to be determined, something that can not be done in a straightforward manner. In our case, the simulated federated environment used made it possible to calculate the silhouette score for evaluation purposes. In a real-life setting, the simplified silhouette score (Hruschka, Castro, and Campello 2004) could be a suitable alternative, as it only calculates distances between datapoints and cluster means, something which can be done on all clients separately.

One of the limitations of this work is the use of only one non-synthetic dataset. Furthermore, even though FEMNIST is split based on author of the digit, it remains a single dataset split up over multiple clients. Various issues can arise when combining multiple datasets, such as batch effects or domain shifts, which is the realistic scenario for federated learning applications. Assessment of our method on more heterogeneous and 'real life' datasets is therefore an important direction for future work.

References

Arthur, David and Sergei Vassilvitskii (2006). *k-means++: The Advantages of Careful Seeding*. Tech. rep.

Caldas, Sebastian et al. (2018). "LEAF: A Benchmark for Federated Settings". In: *CoRR* abs/1812.01097. arXiv: 1812.01097. URL: http://arxiv.org/abs/1812.01097.

Cohen, Gregory et al. (2017). "EMNIST: an extension of MNIST to handwritten letters". In: *CoRR* abs/1702.05373. arXiv: 1702.05373. URL: http://arxiv.org/abs/1702.05373.

Dennis, Don Kurian, Tian Li, and Virginia Smith (2021). "Heterogeneity for the Win: One-Shot Federated Clustering". In.

Hartigan, J.A and M.A. Wong (1979). "Algorithm AS 136: A K-Means Clustering Algorithm". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28, pp. 100–108.

Hou, Ruiqi et al. (2021). "Multi-Party Verifiable Privacy-Preserving Federated k-Means Clustering in Outsourced Environment". In: DOI: 10.1155/2021/3630312. URL: https://doi.org/10.1155/2021/3630312.

Hruschka, E.R., L.N. de Castro, and R.J.G.B. Campello (2004). "Evolutionary algorithms for clustering gene-expression data". In: *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pp. 403–406. DOI: 10.1109/ICDM.2004.10073.

Kumar, Hemant H, V R Karthik, and Mydhili K Nair (2020). "Federated K-Means Clustering: A Novel Edge AI Based Approach for Privacy Preservation; Federated K-Means Clustering: A Novel Edge AI Based Approach for Privacy Preservation". In: *2020 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*. DOI: 10.1109/CCEM50674.2020.00021.

Lee, Geun Hyeong and Soo Yong Shin (Oct. 2020). "Federated Learning on Clinical Benchmark Data: Performance Assessment". In: *Journal of Medical Internet Research* 22.10. ISSN: 14388871. DOI: 10.2196/20891.

Li, Qinbin et al. (2021). "A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection — Enhanced Reader". In: *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1. DOI: 10.1109/TKDE.2021.3124599.

Li, Tian et al. (2020). "Federated Learning: Challenges, methods and future directions". In: DOI: 10.1109/MSP.2020.2975749.

Liu, Yang et al. (June 2020). "Privacy-preserving federated k-means for proactive caching in next generation cellular networks". In: *Information Sciences* 521, pp. 14–31. ISSN: 00200255. DOI: 10.1016/J.INS.2020.02.042.

McMahan, H. Brendan et al. (2017). "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: 54, p. 10.

Sadilek, Adam et al. (Sept. 2021). "Privacy-first health research with federated learning." In: *NPJ digital medicine* 4.1, p. 132. ISSN: 2398-6352. DOI: 10.1038/s41746-021-00489-2. URL: http://www.ncbi.nlm.nih.gov/pubmed/34493770%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC8423792.

Servetnyk, Mykola, Carrson C Fung, and Zhu Han (2020). "Unsupervised Federated Learning for Unbalanced Data; Unsupervised Federated Learning for Unbalanced Data". In: *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. DOI: 10.1109/GLOBECOM42002.2020.9348203.