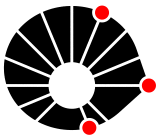

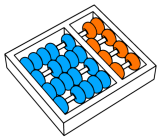

# Seleção de Dados não Ruidosos em Modelos de Aprendizado de Máquina Federado

*P. J. Novais* *T. H. Costa* *F.M. Roberto*
*L. F. Bittencourt*

Relatório Técnico - IC-PFG-23-29
Projeto Final de Graduação
2023 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

# Seleção de Dados não Ruidosos em Modelos de Aprendizado de Máquina Federado

Pedro Jun Novais  Thiago Henrique da Costa  Filipe Maciel Roberto
Luiz Fernando Bittencourt *

**Resumo**

Tendo em vista os progressos recentes no campo da inteligência artificial, o rápido e crescente desenvolvimento da computação móvel e o irrevogável direito à privacidade, modelos de aprendizado federados têm se tornado peça-chave no desenvolvimento tecnológico atual. O objetivo deste projeto é avaliar uma tratativa de dados ruidosos em modelos de classificação de imagens, utilizando a técnica de aprendizado por reforço.

## 1 Introdução

O aprendizado de máquina federado (FL - *Federated Learning*) é uma abordagem de aprendizado de máquina (ML - *Machine Learning*) descentralizada, que permite treinar modelos de forma colaborativa sem compartilhar os dados privados dos participantes, tendo em vista o artigo "*A survey on federated learning: challenges and applications*"(Wen, J., Zhang, Z., Lan, Y. et al (2023)). Essa abordagem surgiu como uma solução referente aos desafios de privacidade, segurança e eficiência que as técnicas tradicionais de aprendizado de máquina centralizado enfrentam ao lidar com grandes volumes de dados distribuídos em diferentes dispositivos.

As principais estratégias de FL seguem a seguinte metodologia: um servidor central coordena o processo de treinamento que é realizado em rodadas (rounds), no início de cada rodada o servidor envia o modelo inicial ou os parâmetros do modelo para um conjunto de clientes, que podem ser dispositivos móveis, sensores etc. Cada cliente treina o modelo localmente com seus próprios dados e envia de volta ao servidor apenas as atualizações do modelo, que são agregadas pelo servidor para obter um modelo global, esse conjunto de passos configura uma rodada completa de treino. O processo se repete até que o modelo atinja um critério de convergência, qualidade ou número de ciclos desejados. Observe a figura a seguir:

*Instituto de Computação - Universidade Estadual de Campinas (UNICAMP), 13083-970 Campinas-SP, Brasil

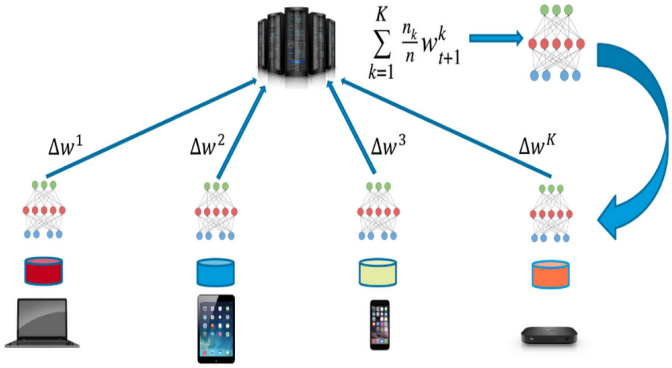


Figura 1: Ilustração do funcionamento de um sistema de Aprendizado Máquina Federado.

Essa técnica tem diversas vantagens, como a preservação da privacidade dos dados, tema amplamente discutido nos últimos anos, em território nacional, desde a sanção da Lei Geral de Proteção de Dados Pessoais (Lei n.º 13.709/2018). Tendo em vista que apenas as atualizações do modelo são enviadas para o servidor ao invés dos dados pessoais que possam estar presentes em cada nó. Além disso, podemos mencionar outros pontos positivos, como a redução do custo de comunicação e armazenamento, a adaptação a cenários dinâmicos e a possibilidade de personalização do modelo.

No entanto, também existem algumas limitações que precisam ser superadas, como: a baixa qualidade de dados que impacta diretamente no desempenho dos modelos, implicando em baixa acurácia ou em alguns casos a não convergência, motivado por esses fatores os campos de seleção de clientes e seleção de dados têm ganhado importância significativa na atualidade.

A seleção de clientes no contexto de FL é o domínio no qual dado um subconjunto de clientes que podem ser utilizados para treino, o servidor seleciona apenas aqueles que possuem dados relevantes ou não ruidosos para atualização do modelo global, essa frente foi explorada em detalhes no artigo "*Client Selection in Federated Learning: Principles, Challenges, and Opportunities*" (Fu, L., Zhang, H., Gao, G., Zhang, M., and Liu, X. (2023)). Por outro lado, seleção de dados no contexto de FL pode ser interpretada como um filtro de dados não relevantes aplicados pelos clientes sobre o dataset de treino, essa abordagem foi utilizada neste trabalho para otimizar o desempenho dos modelos federados quando treinados em dados não confiáveis, a categorização e quantificação dos dados com relação a sua relevância utilizada neste projeto foi baseado no artigo "*Data Valuation using Reinforcement Learning*" (Yoon, J., Arik, S. O., and Pfister, T. (2019)).

# 2 Metodologia

Para mensurar a solução avaliada neste trabalho, a implementação foi dividida na construção de 3 modelos diferentes, sendo o primeiro um modelo tradicional de FL seguindo a estratégia FedAVG, o segundo é um modelo similar ao primeiro, porém, foi treinado usando um conjunto de dados com ruído. Por fim, o terceiro modelo, também foi construído com base no FedAVG, todavia em cada rodada de treino foi aplicado um filtro de relevância de dados (DVRL).

## 2.1 FedAVG (Federated Average)

Como mencionado anteriormente, os modelos de FL implementados seguem a estratégia FedAVG e o algoritmo base de funcionamento tem a seguinte estrutura:

1. Seja i a variável a contadora de rodadas de treino e j o identificador de cada cliente pertencente a esse sistema distribuído, tal que $\{i, j \, \epsilon \, N;\, 1 \leq i \leq n \text{ e } 1 \leq j \leq c\}$, sendo que n representa o número total de rounds e c o número total de clientes, se i = 1 o servidor inicia o modelo global com parâmetros arbitrários, que será representado por $\theta_{0,global}$.

2. No início de cada rodada de treino, o servidor envia os parâmetros do modelo global para os clientes, ou seja, os clientes recebem do servidor $\theta_{i-1,global}$.

3. Os clientes carregam $\theta_{i-1,global}$ em seus modelos locais e utilizam $D_{train,i,j}$ ($D_{train,i,j}$ é a representação do conjunto de dados utilizado para treino pelo cliente j na rodada i) para atualizar o seu modelo local.

4. Finalizado a rodada de treino, cada cliente obtém $\theta_{i,local,j}$ e envia para o servidor $\delta_{i,j} = \theta_{i,local,j} - \theta_{i-1,global}$.

5. O servidor agrega e atualiza o modelo global utilizando os resultados dos clientes, da seguinte forma $\delta_{i,agg} = \frac{1}{c} \sum_{j=1}^{c} \delta_{i,j}$ e $\theta_{i,global} = \theta_{i-1,global} + \delta_{i,agg}$.

6. Por fim, o servidor testa o modelo global atualizado em $D_{test,i}$ (conjunto de dados utilizado para teste na rodada i), então é obtida a acurácia e a perda do modelo global.

7. As etapas de 2 a 6 se repetem até um número de rodadas predefinido seja executado ou estado de convergência definido aceitável previamente.

Nesse projeto foi utilizado o dataset CIFAR10, que compila imagens de 32×32 pixels de dez categorias diferentes, por padrão o tamanho $D_{train,i,j}$ foi fixo em 5.000 imagens, para todos os i e j, e o tamanho utilizado do $D_{test,i}$ foi de 10.000 imagens. Além disso, os testes foram realizados com c = 2 e n = 100, e todos os modelos executaram 10 epochs com uma taxa de aprendizado de 0,01.


Figura 2: Exemplos de imagens do *dataset* CIFAR10.

## 2.2 Ruído

Existem diversos tipos de ruídos possíveis no universo de ML, e eles são divididos em dois grandes grupos:

1. Ruído de rótulo (*Label noise*): esse caso consiste em erros ou inconsistências no processo de rotulação de dados, podendo ser causado por diversos motivos, os mais comuns são erros humanos no processo, critério ambíguo de criação de rótulos e manipulação intencional ou maliciosa de rótulos.

2. Ruído de características (*Feature noise*): esse evento está relacionado com variações não desejadas e prejudiciais nos dados. Por exemplo: dados incompletos, formatações erradas ou *outliers*.

Nos modelos foram aplicados ruídos de rótulos nos dados de treino, implementado da seguinte forma: seja $D_{train,total,j} = \bigcup_{i=1}^{n} D_{train,i,j}$ o conjunto total de dados utilizado por um cliente arbitrário para treino e $\rho_{noise}$ a representação do percentual ruído aplicado sobre $D_{train,total,j}$ tal que $\rho_{noise} \epsilon R$ e $0 \leq \rho_{noise} \leq 1$, é definido o conjunto $D_{pre-noise,j}$ formado por imagens escolhidas aleatoriamente de $D_{train,total,j}$ de forma que $|D_{pre-noise,j}| = |D_{train,total,j}| \cdot \rho_{noise}$, toda imagem pertencente a $D_{pre-noise,j}$ tem seus rótulos trocados formando $D_{noise,j}$, então é obtido $D_{train-noise,total,j} = (D_{train,total,j} - D_{pre-noise,j}) \cup D_{noise,j}$ sendo o novo conjunto total de dados que será utilizado para treino pelo cliente j antes de ser dividido em tamanhos iguais por rodadas de treino. Em todos os modelos que possuíam ruídos, foi utilizado $\rho_{noise} = 0.30$, já que esse percentual é significativo para impactar o desempenho dos modelos, mas não grande o suficiente para impedir a convergência destes.

## 2.3 DVRL (Data Valuator using Reinforcement Learning)

DVRL é o modelo que atribui uma probabilidade de relevância em cada dado de entrada nas rodadas de treino, o DVRL foi treinado separadamente antes do experimento e teve seus parâmetros carregados por cada cliente antes da primeira rodada de treino, nesse tópico será abordado como esse modelo foi constituído, treinado e sua integração no modelo federado. A implementação do DVRL seguiu os conceitos propostos no artigo "Data Valuation using Reinforcement Learning"(Yoon, J., Arik, S. O., and Pfister, T. (2019)).

### 2.3.1 Funcionamento e treino DVRL

O funcionamento do DVRL é uma combinação de dois modelos, um classificador e outro que estima o valor de relevância e encontra a distribuição de probabilidade dos dados, essa

combinação é muito poderosa e possibilita que o modelo consiga ranquear a importância dados os quais são expostos.

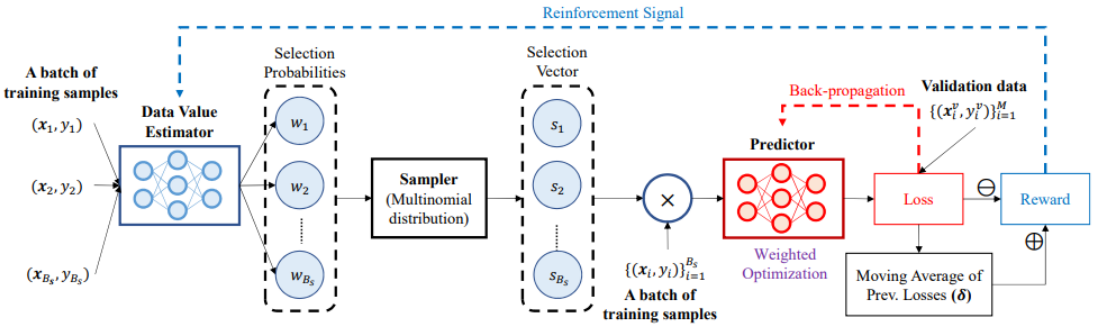


Figura 3: Diagrama de treino referente ao modelo DVRL. Créditos: Yoon, J., Arik, S. O., and Pfister, T. (2019).

O treino dos modelos DVRL seguem o conjunto de passos abaixo:

1. Um batch de imagens $B_i$ passa pelo modelo estimador dos valores das amostras.

2. O modelo anterior retorna o vetor $W = (h(d_1), h(d_2), ..., h(d_k))$, sendo que $(d_1, ..., d_k) \epsilon B_i$. W corresponde ao vetor de probabilidade de seleção dos dados, e satisfaz uma distribuição multinomial, para um problema de classificação.

3. O vetor W passa por um amostrador que, aplica uma tranformação de tal forma que se atribui o valor 1 para valores acima de um limite e 0 para valores menores ou iguais a esse limite, ou seja, é retornado o vetor $S = (s_1, ..., s_k)$, tal que $\{\forall d_i \epsilon B_i; (s_i = 0,\ se\ h(d_i) \leq l)\ ou\ (s_i = 1,\ se\ h(d_i) > l)\}$.

4. Os dados que tiverem valor correspondente a 1 no vetor S, ou seja, $s_i = 1$, serão utilizados para treino no modelo classificador.

5. O modelo classificador realiza o treino, utilizando convencionalmente o algoritmo de gradiente descendente, com os dados que satisfazerem a condição anterior.

6. A perda do modelo classificador é comparada com uma média móvel das perdas calculadas para os batches anteriores, então é obtida a recompensa.

7. O primeiro modelo que é o estimador dos valores das amostras atualiza seus parâmetros com base em um sinal de reforço guiado pela recompensa.

8. Os passos de 1 a 7 se repetem para todos batches pertencentes ao conjunto de dados de treino e para uma quantidade de epochs definida.

### 2.3.2 Integração com o modelo federado

Com os parâmetros carregados pelos clientes, a integração do modelo que atribui valores aos dados de entrada do DVRL com o ambiente distribuído, foi relativamente simples e seguiu a lógica explicitada no parágrafo abaixo. É importante ressaltar que o modelo classificador do DVRL não foi carregado pelos clientes, visto que ele será substituído pelo modelo local.

Seja $\alpha_{irrelevant\ data}$ o percentual de dados não relevantes que devem ser removidos da rodada de treino dos modelos federados, o modelo DVRL atribui uma probabilidade em cada ponto de dado pertencente a $D_{train-noise,i,j}$ (conjunto de imagens com ruído utilizado para treino pelo cliente j na rodada i), então é formado o conjunto de dados não relevantes que devem ser excluídos $D_{irrelevant\ data,i,j}$, tal que $D_{irrelevant\ data,i,j}$ é formado pelos $|D_{train-noise,i,j}| \cdot \alpha_{irrelevant\ data}$ menos relevantes pertencentes a $D_{train-noise,i,j}$, é finalmente obtido o novo conjunto de dados que será utilizado pelo cliente j na rodada i de treino do modelo local $D_{train-noise-dvrl,i,j} = D_{train,i,j} - D_{irrelevant\ data,i,j}$. Vale pontuar que por simplicidade na implementação foi utilizado $\alpha_{irrelevant\ data} = 0.10$.

## 3 Trabalhos relacionados

A valoração de dados (*data valuation*) é um tema fundamental em aprendizado de máquina que visa quantificar o valor de cada amostra de dados para uma determinada tarefa. Tendo múltiplos casos de uso importantes, como construir *insights* sobre a tarefa de aprendizado, adaptação de domínio, descoberta de amostras corrompidas e aprendizado robusto. No entanto, a valoração de dados requer métodos adaptativos e específicos para a tarefa.

Neste relatório, estudamos o *framework* de meta-aprendizado que denominamos *Data Valuation using Reinforcement Learning*. O DVRL está relacionado a trabalhos existentes na literatura, alguns deles listamos a seguir:

1. ***Data valuation***: Existem vários métodos propostos para medir o valor dos dados, como *leave-one-out* (LOO) (Sammut e Webb, 2011), funções de influência (IF) (Koh e Liang, 2019), *Data Shapley* (Ghorbani e Zou, 2019). Esses métodos são baseados em diferentes princípios e suposições, como a contribuição marginal de cada amostra de dados, a sensibilidade dos parâmetros do modelo ou o efeito causal da remoção de dados. No entanto, esses métodos têm algumas limitações, como alta complexidade computacional, problemas de escalabilidade ou dependência do modelo inicial. O DVRL supera essas limitações usando uma abordagem de meta-aprendizado que se adapta à tarefa alvo e aprende os valores dos dados em conjunto com o modelo preditor.

2. ***Meta learning***: Meta-aprendizado é um paradigma que visa aprender com múltiplas tarefas e generalizar para novas tarefas. Existem diferentes tipos de métodos de meta-aprendizado, como baseados em otimização (Finn et al., 2017) e baseados em métrica (Vinyals et al., 2016). O DVRL é um método de meta-aprendizado baseado em modelo que usa um estimador de valor de dados como meta-aprendiz e um modelo preditor como aprendiz base. O DVRL difere de outros métodos de meta-aprendizado baseados

em modelo por não usar um módulo de memória ou uma rede recorrente, mas sim usar um algoritmo de aprendizado por reforço para treinar o meta-aprendiz.

3. ***Reinforcement learning***: Aprendizado por reforço é um *framework* que permite a um agente aprender com suas próprias ações e recompensas. Existem diferentes tipos de métodos de aprendizado por reforço, como baseados em valor (Mnih et al., 2015). O DVRL é um método de aprendizado por reforço baseado em política que usa um estimador de valor de dados como agente e uma recompensa do conjunto de validação como função de recompensa. O DVRL difere de outros métodos de aprendizado por reforço baseados em política por não usar uma política estocástica, mas sim usar uma política determinística que produz valores binários para cada amostra de dados.

# 4 Resultados

Observamos, durante nossas análises, que a acurácia do modelo, que se utiliza do FedAVG sem ruído e utilizando 100 rounds de treino, aumentou de maneira vertiginosa nos primeiros rounds de treino. Além disso, próximo à trigésima rodada podemos observar que o modelo busca o estado de convergência, ou seja, a assertividade do modelo varia muito pouco a cada nova rodada de treino a partir desse ponto. Outrossim, a metodologia FedAVG acrescida de 30% de ruído de rótulos começa estabilizar por volta da sexagésima rodada. Também é possível pontuar que o modelo avaliado, o FedAVG com ruído e filtro de dados não relevantes, atinge o mesmo patamar de estabilidade por volta da quadragésima rodada.

Vale ressaltar quanto antes atingir o estado de estabilidade da acurácia é melhor para o modelo. Podemos observar a seguir os gráficos comparativos entre os modelos:

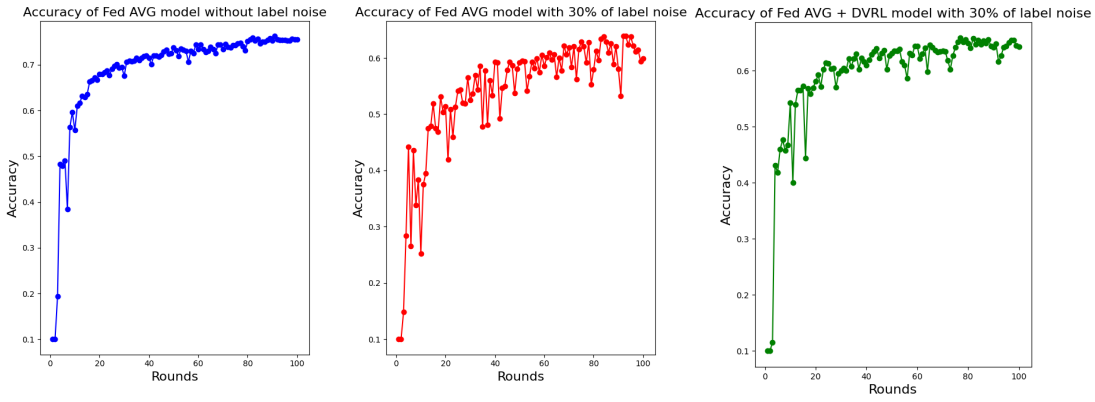


Figura 4: Acurácia dos modelos por rodada de treino.

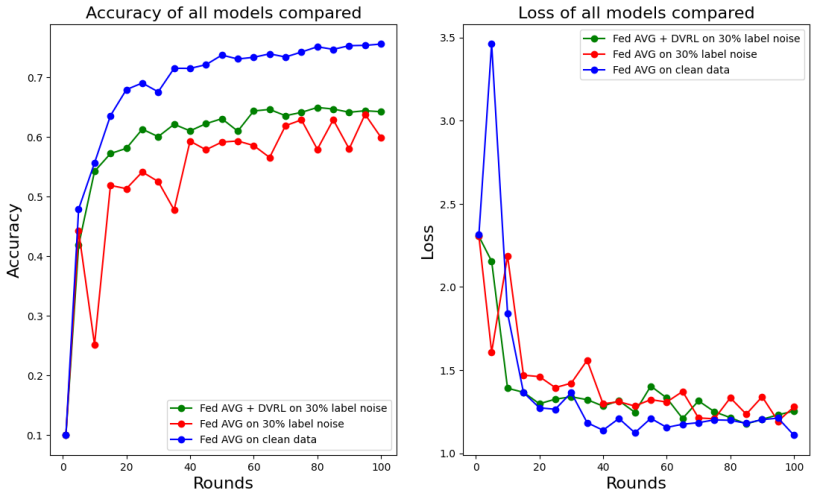


Figura 5: Comparativo entre os modelos.

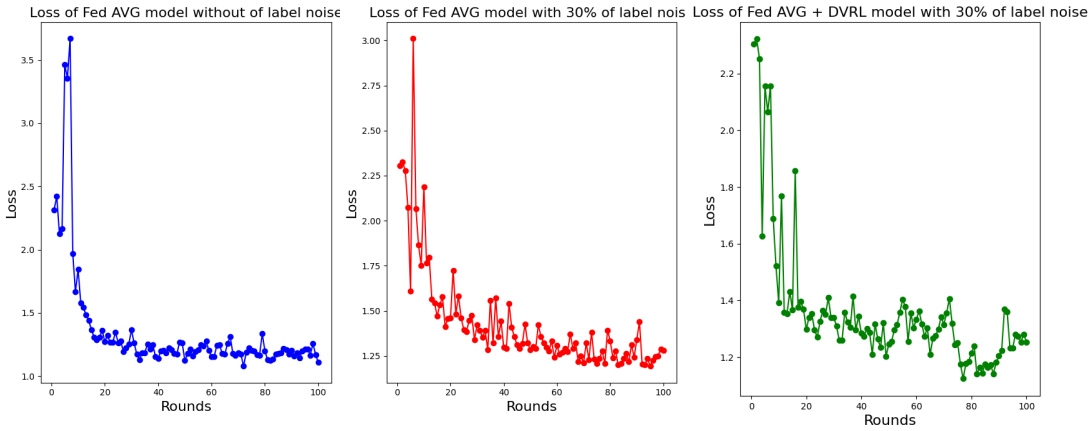


Figura 6: Função de perda dos modelos por rodada de treino.

Ao realizar o comparativo dos modelos, percebe-se dois extremos: o modelo sem ruído possui a melhor assertividade com a menor volatilidade entre rodadas, como era esperado; já o modelo com ruído e sem implementação do DVRL, possui o pior quadro dentre os três analisados, tendo em vista suas oscilações e menor acurácia média. Somado a isso o modelo com ruído e com implementação do DVRL, detêm uma acurácia maior e oscila muito menos que a implementação sem DVRL com ruído, isso acentua-se principalmente nos primeiros rounds.

Sendo assim, podemos concluir que em um ambiente onde a quantidade de dados não confiáveis detém uma parcela significativa do total, a implementação do modelo DVRL se mostrou benéfica. Haja vista que nos leva a um aumento da acurácia e diminui a sua volatilidade, principalmente nas rodadas iniciais. Porém, seu desempenho é consideravelmente

pior que o modelo sem ruído, não conseguindo mitigar na totalidade os dados não confiáveis, apesar do comportamento das curvas de performance e perda serem relativamente parecidos.

# 5 Conclusão

Neste relatório, foi avaliada, testada e analisada a implementação de um filtro de dados ruidosos em ambientes de aprendizado de máquina distribuídos, o qual atribui valores de relevância aos dados de treino, estimando sua distribuição de probabilidade de ser usado em cada rodada. Isso foi possível, pois os clientes receberam e acoplaram aos seus modelos locais o *Data Valuator using Reinforcement Learning*. O DVRL treina um estimador de valor de dados usando um sinal de reforço da recompensa obtida em um pequeno conjunto de validação, que reflete o desempenho na tarefa alvo.

Apesar deste projeto contemplar apenas modelos de classificação de imagens, o DVRL pode lidar também com outros tipos diferentes de conjuntos de dados e tarefas, como regressão e previsão de séries temporais. Sendo assim, constatamos que é uma solução interessante, flexível e de fácil implementação, que pode ser moldada dependendo da sua aplicação para se obter o máximo desempenho e eficácia, no sentido de mitigação de ruídos em aprendizado de máquina federado, trazendo diversos benefícios, sendo o principal deles maior assertividade dos modelos.

## Referências

[1] Nagalapatti, L., Mittal, R. S., and Narayanam, R. (2022). *Is Your Data Relevant?: Dynamic Selection of Relevant Data for Federated Learning. AAAI Conference on Artificial Intelligence*, 36(7), 7859-7867. doi.org/10.1609/aaai.v36i7.20755

[2] BRASIL. Lei nº 13.709, de 14 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais. Diário Oficial da União, Brasília, DF, 15 ago. 2018.

[3] Fu, L., Zhang, H., Gao, G., Zhang, M., and Liu, X. (2023). *Client Selection in Federated Learning: Principles, Challenges, and Opportunities*. arxiv.org/abs/2211.01549

[4] Yoon, J., Arik, S. O., and Pfister, T. (2019). *Data Valuation using Reinforcement Learning*. arxiv.org/abs/1909.11671

[5] Wahab, O. A., Mourad, A., Otrok, H. and Taleb, T. (2021). *Federated Machine Learning: Survey, Multi-Level Classification, Desirable Criteria and Future Directions in Communication and Networking Systems. IEEE Communications Surveys and Tutorials*. 10.1109/COMST.2021.3058573

[6] Magnusson, M., Andersen, M. R., Jonasson, J., and Vehtari, A. (2019). *Bayesian leave-one-out cross-validation for large data*. arxiv:1904.10679

[7] Koh, P. W., and Liang, P. (2017). *Understanding black-box predictions via influence functions. In Proceedings of the International Conference on Machine Learning* (pp. 1885–1894). https://proceedings.mlr.press/v70/koh17a

[8] Sammut, C., and Webb, G.I. (2011). *Leave-One-Out Cross-Validation. In Encyclopedia of Machine Learning*. Springer, Boston, MA. doi.org/10.1007/978-0-387-30164-8_469

[9] Finn, C., Abbeel, P., and Levine, S. (2017). *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*. arxiv:1703.03400

[10] Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. (2016). *Matching Networks for One Shot Learning*. arxiv:1606.04080

[11] Mnih, V., Kavukcuoglu, K., Silver, D. et al (2015). *Human-level control through deep reinforcement learning*. Nature 518, 529–533. doi.org/10.1038/nature14236

[12] Wen, J., Zhang, Z., Lan, Y. et al (2023). *A survey on federated learning: challenges and applications*. Int. J. Mach. Learn. and Cyber. 14, 513–535. https://doi.org/10.1007/s13042-022-01647-y