Confidential Maternal Health Guardian Project Report

Kaan Turkmen, Can Usluel, Berke Can Rizai, Eren Baris Bostanci

January 18, 2023

1 Introduction

The health of women during pregnancy, childbirth, and the postnatal period, referred to as maternal health, is crucial in ensuring that both the woman and her baby reach optimal health and well-being. Despite progress in recent years, a significant number of women still die from complications related to pregnancy and childbirth, a statistic that is considered unacceptable. Some of the main components that identify the risk to maternal health are age, blood pressure, blood sugar, body temperature, and heart rate. In our project, we aim to predict the maternal health risk of the user by using these parameters in our machine learning model. In addition, we aim to inform the user about the statistics of our data set, meanwhile, protecting the privacy of the users.

2 Architecture and Technologies

Our first aim while developing CMHG application is to make it accessible to everyone by hosting front end in the AWS. However, due to increased size of the ML dependencies, we did not want to upgrade our instance type to the higher one because of the cost. Thus, architecture will be the same with the below figure, but it will not be on the cloud.

In terms of technologies, we have used PostgreSQL for database, Python for internal back end and Java Spring Boot for the main back end. Furthermore, we have used Typescript and React for our front end. Those technologies use REST to communicate with each other. If we were in the AWS, we would have blocked internal back end's specified port from the security groups, however, that is not the case in the local development. We dockerized all the components of the project to create compact design and ease of use while deploying. We will be covering how can you deploy the application with one command in the Setup section.

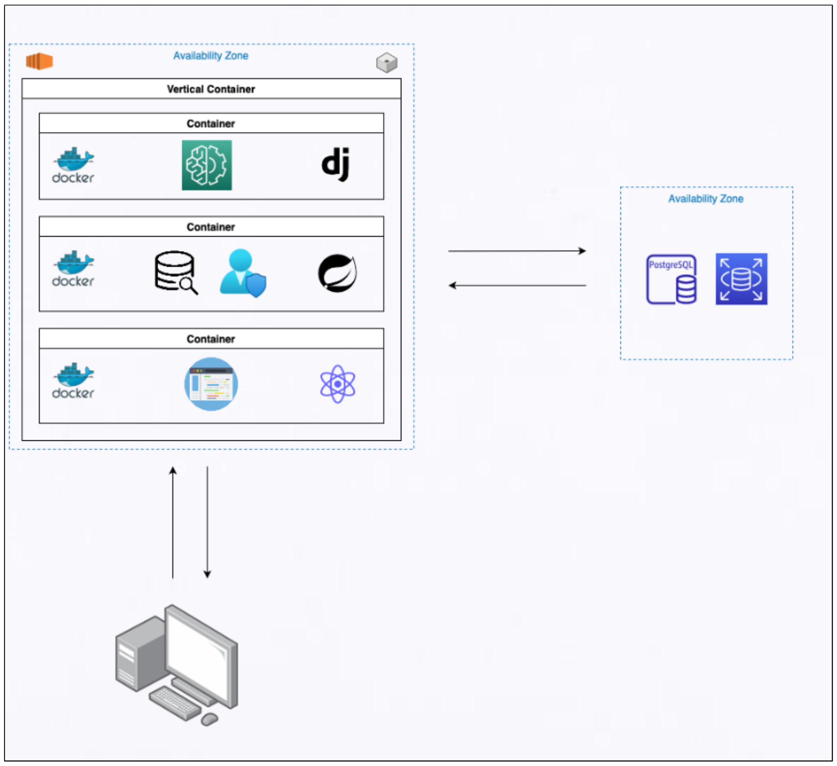


Figure 1: Proposed Architecture.

3 Setup

We are including source code and model trainings in the zip file. However, we highly recommend you to use docker compose[1] to run the project. Please keep in mind that, project is around 3-4 Gigabytes because of the Python machine learning dependencies.

- Download Docker & Docker Compose
- From terminal change directory to the docker-compose.yaml file location which is in the most outer directory of the project folder.
- Type docker compose up (or docker-compose up) to run the project.

It takes longer to deploy if you are using ARM chipped Macbook, because docker images are built using amd64 architecture. Even though it takes multiple minutes to deploy, everything works fine in ARM chips as well.

[1]Docker Compose is a tool for defining and running multi-container Docker applications.

4 Data Set

We selected ”Maternal Health Risk Data Set” [1] for our project. The main reason is, This data set is created by collecting the data of patients in different hospitals. Thus, the models trained with this data can make better predictions, and we can apply data privacy techniques to real-life data. Moreover, It is large enough to train a good machine learning model and large enough to make analysis via statistical queries. The structure and format of the data set are well-suited for the tasks that the application will be performing, and it contains relevant information that will allow the machine learning models to generate reasonable results. Overall, we selected the ”Maternal Health Risk Data Set” for its compatibility with the tasks of the application, its size and complexity.

5 Features

5.1 Noisy Query

For this part, we are not using a synthetically produced dataset. By using the original dataset, we are adding laplace noise[2] to the result and returning the noisy one. By doing that, we are hiding the exact details of the users.

5.2 Differentially Private Estimation

Here, in order to handle membership inference attacks[3], we have created whole dataset with synthetic data and in order to make sure we do not lose any functionality, we have kept the distribution of the data in mind.

For this, we have fitted several kernels[4] to each of the features for each of the classes and from the validated results, saved the best fitted kernel. We then sample from these kernels 1000 times according to prior distribution of each of the classes. To better visualize the effect of synthetic vs original data, we have run vanilla RandomForest from the python sklearn library for 100 times each and plotted the results in a box plot.

[2]Laplace noise is a type of random noise that is added to a dataset to protect the privacy of the individuals in the dataset.

[3]A membership inference attack is a type of attack on a machine learning model where an attacker attempts to determine whether a specific individual’s data was used to train the model.

[4]In the context of machine learning, a kernel is a mathematical function that is used to transform data into a higher dimensional space, making it possible for a machine learning algorithm to learn more complex patterns.

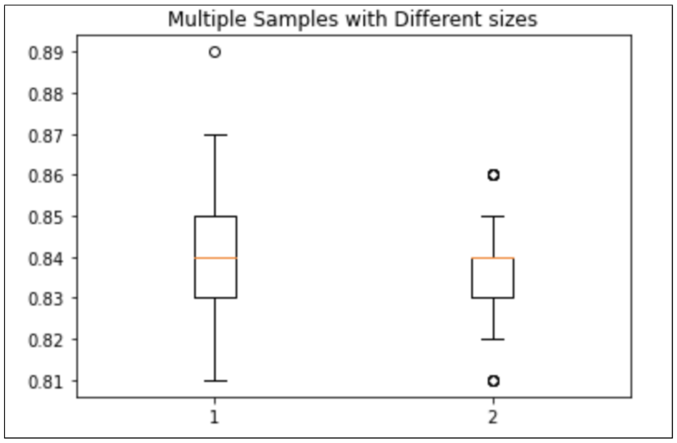


Figure 2: Comparison of RF on original vs synthetic data.

We visualized one of the dimensions of the data and on the left side, we see the original distribution and on the right, we have synthetic data distribution.

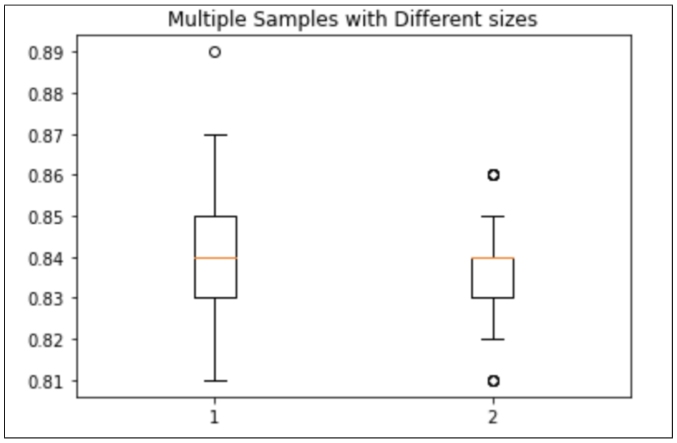


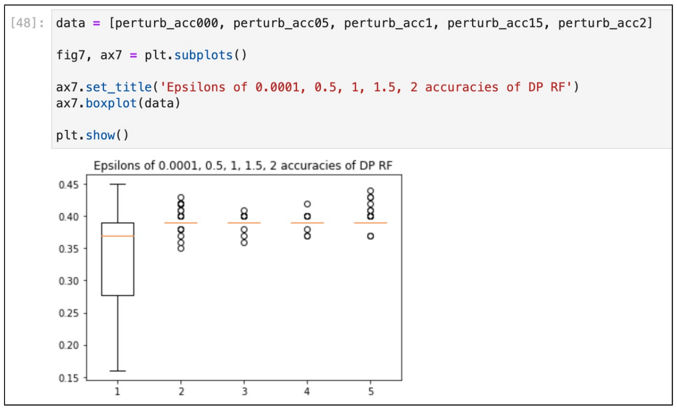
Figure 3: Distributions of synthetic vs original data in one dimension.

After taking this synthetic data, we have trained several models both DP and non DP such as DP random forest, random forest, SGD, DP SGD.

In the DP Random Forest, PermuteAndFlip mechanism is employed. By the authors of the paper [2]: A new mechanism for differentially private selection it is explained as. "For each item, it flips a biased coin, and returns that item if the coin comes up heads. The probability of heads is an exponential function of the quality score, which encourages the mechanism to return results with higher quality scores. The mechanism is guaranteed to terminate with a result because if qr = q∗, then the probability of heads is 1."[2].

We have compared different models in terms of accuracy to see how the privacy, synthetic data and other elements effect the utilization of the model. We need to have trade-off between the privacy and the accuracy since too accurate model could lead to problems in privacy but if the model is private but predicting power is too low, our maternal risk tool would be rendered useless.

Here is a comparison of different epsilon values on DP random forest where number of estimators are 20 and all other parameters are same. This was made with 100 runs and accuracies are plotted on box-plot. We can see that very low epsilon has



```
[48]: data = [perturb_acc000, perturb_acc05, perturb_acc1, perturb_acc15, perturb_acc2]

      fig7, ax7 = plt.subplots()

      ax7.set_title('Epsilons of 0.0001, 0.5, 1, 1.5, 2 accuracies of DP RF')
      ax7.boxplot(data)

      plt.show()
```

Figure 4: Accuracy comparison of DP Random Forest with different epsilons over 100 runs.

5.3 Differential Private Stochastic Gradient Descent

As an alternative prediction model, we implemented a neural network and to satisfying privacy, we implemented (ϵ, δ) differential private stochastic gradient descent as an optimizer. We used PyTorch library to build our neural network in python. (ϵ, δ) differential privacy, ϵ is the privacy budget and lower epsilon values provide better privacy. δ is the probability of the information accidental being leaked. δ must be small to ensure the privacy. In our implementation we give 10^{-8} as an δ. To implement (ϵ, δ) differential privacy, we followed 4 steps [3].

- Compute Gradient
- Clip Gradient
- Add Noise
- Descent

First we compute gradients as in vanilla SGD. Secondly we clipped the gradient before adding noise to prevent the gradients becoming too large, which can cause oscillation problems and instability during the training process. In noise adding part to satisfy (ϵ, δ) differential privacy, we scaled noise with Gaussian Mechanism. The scaling formula is:

$$\sigma = \sqrt{2 \log \frac{1.25}{\delta}} / \epsilon$$

We trained 10 models for different epsilon values and take the average of them.

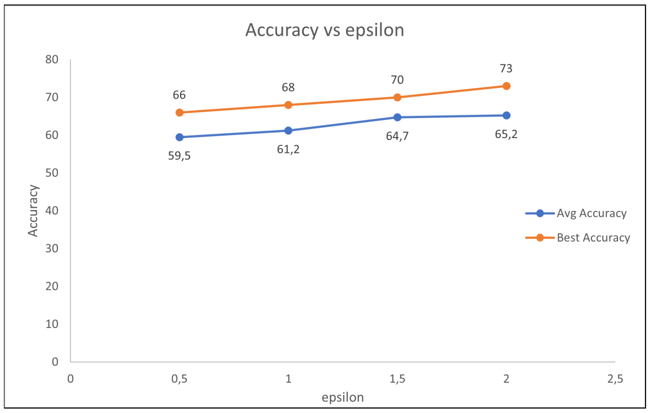


Figure 5: DP-SGD Accuracy vs ϵ.

As we can see in the figure 5, for the smaller epsilon values accuracy is smaller since the privacy is high. Also for the higher epsilon values the accuracy is higher due to lower privacy as we expected.

We also trained 10 models more. This time we used constant epsilon value 2.0 with different delta values $[10^{-8}, 10^{-5}, 10^{-2}, 1]$ and take the average of the 10 runs to examine the effect of the δ.

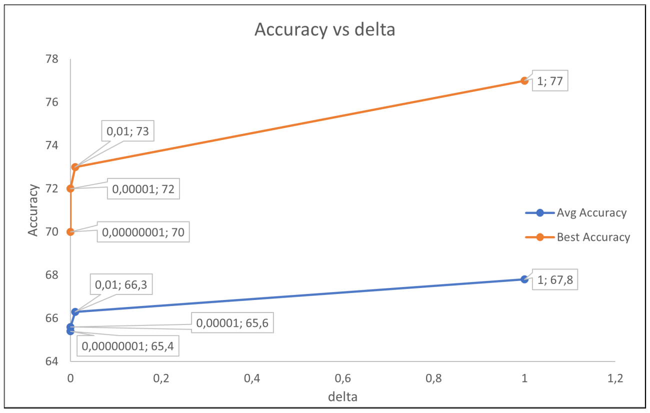


Figure 6: DP-SGD Accuracy vs δ.

We can see in the figure 6, for higher δ values the accuracy increase since the probability of the information being leaked is higher. When the δ is 1, the information 100% leaked and the avg accuracy is as high as before the applying (ϵ, δ) differential privacy. For the privacy purpose, we do not let the users select δ value from user interface and use 10^{-8} as constant value for the models that used in application.

We also trained non-differential private version of models and the accuracy was 75%. However we do not let users to access the vanilla SGD version of the model. Also, in the application we exported trained models to speedup the run time. We saved models by selecting best models from 4 trained version of each epsilon values and vanilla SGD version. We load these models when the python backend is starting up. The best model selection part is done in a jupyter notebook, which can be found at the python backend source code under notebooks directory.

5.4 JSON Web Token

In authentication we used JWT, and in authorization we used Spring Security. In our frontend, we are writing this access and refresh tokens to cookies to improve our security. In the backend, we are signing the tokens with the 10 and 30 minute expiration data using pepper-like system to add one more layer of security.

5.5 BCrypt Password Encoder

While registering the user to the website, we used BCrypt Password Encoder with default round 10 to store password of users more secure. We could have increased the number of rounds, however, as mentioned in the class, it adds slowness to the authentication system as well. We did not want to trade-off user experience with more security.

5.6 Database Security

We seperated entity classes from responses to hide our database architecture from the final users. By doing that, we secured our both backend and database. Thus, we added one more layer privacy to the user data.

References

[1] "Maternal health risk data set." [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Maternal+Health+Risk+Data+Set

[2] R. McKenna and D. Sheldon, "Permute-and-flip: A new mechanism for differentially private selection," 2020.

[3] M. Rathi, “Deep learning with differential privacy.” [Online]. Available: https://mukulrathi.com/privacy-preserving-machine-learning/deep-learning-differential-privacy/