A computational resource-friendly prediction system for health condition

Wenjun Huang
36885180
wenjunh3@uci.edu

Ziyu Wang
12726200
ziyuw31@uci.edu

Yang Liu
88889586
yangl73@uci.edu

1 Introduction

It is increasingly recognized that the management of hyperglycemia in hospitalized patients has a significant bearing on the outcome, in terms of both morbidity and mortality. This recognition has led to the development of formalized protocols in the intensive care unit (ICU) setting with rigorous glucose targets in many institutions. However, the same cannot be said for most non-ICU inpatient admissions. Rather, anecdotal evidence suggests that inpatient management is arbitrary and often leads to either no treatment at all or wide fluctuations in glucose when traditional management strategies are employed. Therefore, non-ICU inpatients highly desire to manage their hyperglycemia with the help of easier-use tools. These years, the rapid development of machine learning enlighten a potential way to solve this problem. However, training a machine learning model needs tons of data. The explosively increasing amount of data introduces a challenge to the model training, i.e. the insufficient computational resources in the hospital's local machine.

In this project, we developed a distributed computational resource-friendly hyperglycemia management system, which alleviates this dilemma. It uses a few computation-limited machines and generates personalized advice by using some desensitized data. There are 1 master and 2 workers in our system that implement training tasks. Data collected from each hospital are uploaded to the central master and stored in Hadoop Distributed File System(HDFS)[1] in the master and 2 workers. During the training, the master distributes tasks to workers and sends the model to HDFS system and all hospitals after collecting the results from the workers.

2 Objective

The main objective of the project is to deploy a machine learning model in a distributed computing environment with a focus on hyperglycemia management. Compared with training the model locally, our system saves the computational resources of local machines. It enables users to predict their readmission by inputting health parameters on a user interface. When doing the prediction, the user is asked to provide a few health parameters related to hyperglycemia. After this, the parameters are sent to the central master for inference. The prediction is then sent back and shown on the interface.

The deliverable of our project includes the code of our system and a report demonstrating our ideas, implementations, and experiment results.

3 Architecture

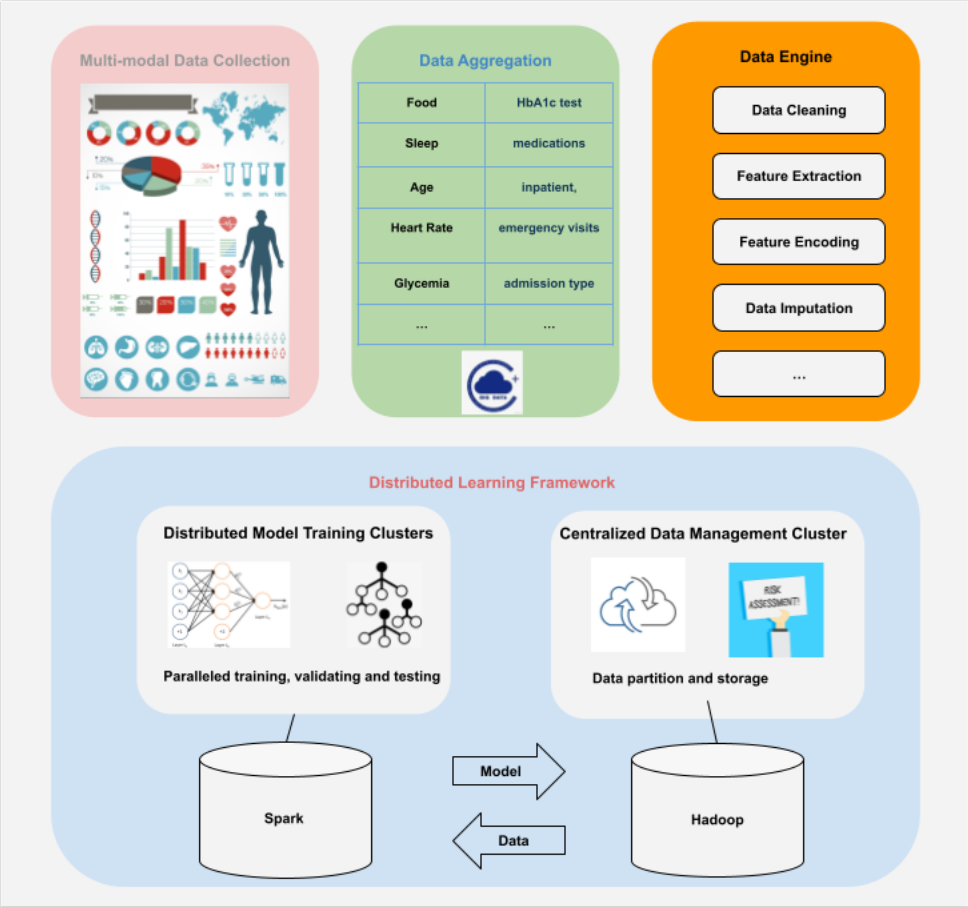


Figure 1: A healthcare distributed system framework

As shown in Figure 1, the architecture contains two components: Distributed Model Training Cluster and Centralized Data Management Cluster. Both of cluster is consisted of three servers.

The role of Distributed Model Training Clusters is to train a model which is used for patient readmission prediction based on the spark framework. The role of the Data Storage Cluster is to store the training data and built a model based on the Hadoop framework.

The running procedures of the whole system are:

- Start Hadoop HDFS cluster and Spark cluster.
- Upload training data to HDFS cluster.
- Submit model training task to Spark cluster.
- Spark cluster retrieves data from the HDFS cluster and starts to train the model.
- Spark cluster saves the model to the HDFS cluster.
- Client will retrieve the model from the HDFS cluster when users send a prediction request.

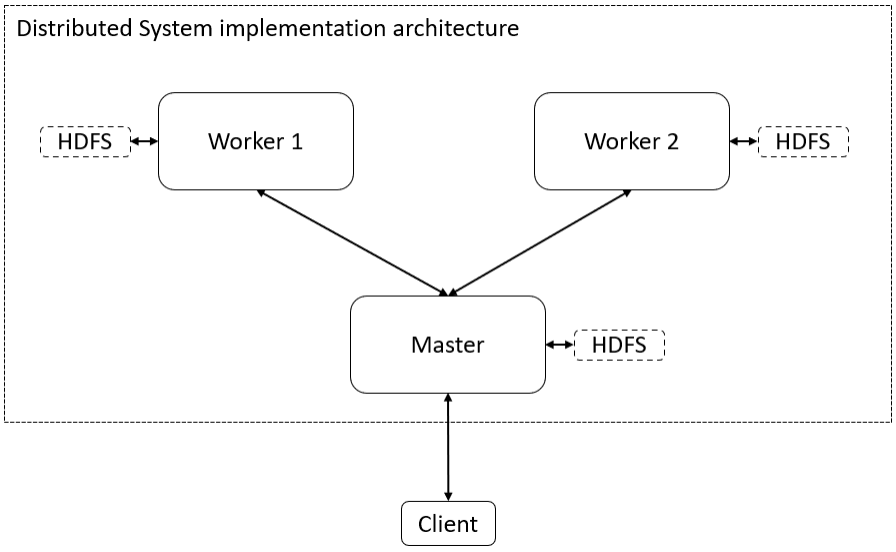


Figure 2: Implementation Architecture

As shown in Figure 2, in real implementation, we set the two clusters in three servers to achieve the framework shown in Figure 1. Hadoop and Spark frameworks are independent of each other. When date and model information is uploaded to the HDFS system, the Hadoop framework will store these data distributively in three servers. Also, when the Spark framework starts to train the model, it will also retrieve data from the HDFS system in three servers and distribute the training task to three servers.

4 Approach

4.1 Dataset

The diabetes 130-US hospital dataset, introduced by Strack et al. (2014) [2], is an invaluable resource for analyzing the clinical care statistics of 130 US hospitals over a span of ten years. This dataset

contains information on inpatient diabetic encounters, where the duration of the stay ranges from one to fourteen days. The dataset includes several essential attributes such as patient number, race, gender, age, admission type, length of stay, number of lab tests, number of medications, emergency visits in the previous year, and more. The dataset's target class comprises three labels: patients with no readmission, patients with readmission within 30 days, and patients with readmission after 30 days. This dataset's comprehensive and diverse attributes make it an invaluable resource for developing predictive models and analyzing the factors that contribute to diabetes hospital readmissions.

4.2 Data Engine

We developed a data engine to facilitate data analysis, which involved crucial steps such as data cleaning, feature extraction, and encoding. To ensure data accuracy, we removed duplicates, handled missing values, and converted data to categorical values. We then identified relevant features by analyzing their statistical relationship with the target label and dropped any unwanted ones to improve the dataset quality. Finally, we converted categorical features to numerical values to ease analysis. By following these preprocessing techniques, we obtained a reliable dataset suitable for predictive modeling and generating insights into the target variable.

4.3 Distributed Training

Our distributed training framework uses Spark [3], Hadoop, and Amazon AWS EC2 VMs. The framework is designed to distribute the training of a machine learning model across multiple VMs to improve scalability and reduce training time with limited computational resources. It consists of two types of clusters:

- Distributed Model Training Clusters: These clusters are responsible for performing the actual model training. They consist of multiple VMs, each running a Spark instance. The VMs are networked together to share the training workload and coordinate the training process. The training tasks are partitioned among the VMs, and each VM works on a subset of the tasks.
- Centralized Data Management Clusters: These clusters are responsible for managing the data used in the training process. They also consist of multiple VMs, each running a Hadoop instance. The training data is stored in a distributed file system, such as HDFS, and is partitioned across the VMs in the data management clusters. The data management clusters partition the data and distribute it to the model training clusters. After the model training is complete, the trained models are aggregated by the data management clusters and stored in a centralized location.

During the training process, the data management clusters communicate with the model training clusters to coordinate the training process. The model training clusters request training data from the data management clusters, which then provide the appropriate data partitions. After the model training is complete, the trained models are sent back to the data management clusters, which aggregate them

into a single model. By using this framework, the training process can be performed on a larger scale, allowing for more complex models and larger datasets. The distributed nature of the framework also allows for faster training times, as the workload is shared across multiple VMs.

4.4 Model as Service Web Application

As part of our project into developing a smart healthcare application, we created a user-friendly web platform to provide personalized risk predictions to users. To make the trained model production-ready, we developed the platform using Flask and hosted it on centralized data management clusters, ensuring easy accessibility with a single click. Our web application allows users to input their medical information and receive risk predictions tailored to their unique circumstances. The user interface is designed to be simple and clear, allowing users to quickly and easily access the information they need.

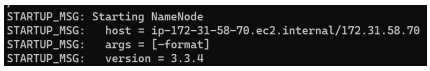
By combining distributed training with a user-friendly web platform, we have developed an innovative smart healthcare application that provides cutting-edge predictive analytics to users. This technology has the potential to revolutionize the healthcare industry by empowering individuals to take control of their health and make informed decisions.

5 Experiment

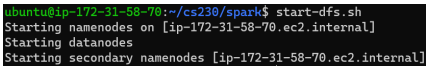
We trained the model in the distributed system environment. The distributed system consists of one master and two workers. All the machines are on Ubuntu 22.04 platform with an 8 GB volume. Instead of having 1 virtual CPU like two workers, the master has 2 virtual CPUs.

After starting the distributed cluster, we reset our Hadoop by executing the command **hadoop namenode -format**. Then we started the data file system with the command **start-dfs.sh**. The dataset was moved to a "data" directory in HDFS by commands **hdfs dfs -mkdir /data** and **hdfs dfs -put ∼/cs230/spark/data/diabetic_data.csv /data/**. All Hadoop daemons, the name node, data nodes, the job tracker, and task trackers are started by **hadoop start-all.sh**. The model is trained by executing **spark-submit –total-executor-cores 3 –master spark://172.31.58.70:7077 model.py**. The argument **–total-executor-cores** determines the number of cores used during the training. The snapshot of each step is shown in Fig. 3.

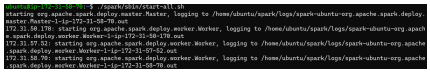
The evaluation results are shown in Fig. 4. Fig. 4a demonstrates the confusion matrix of our model. Label '1' and '0' represent the existence of the risk, respectively. The numbers on the diagonal are the number of correct predictions, the other numbers are the number of misclassifications. Fig. 4b shows the ROC (receiver operating characteristic curve) curve [4] and the AUC score of our model. A ROC curve is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True Positive Rate (TPR), and False Positive Rate (FPR). It plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. AUC stands for



(a) Executing command **hadoop namenode -format**



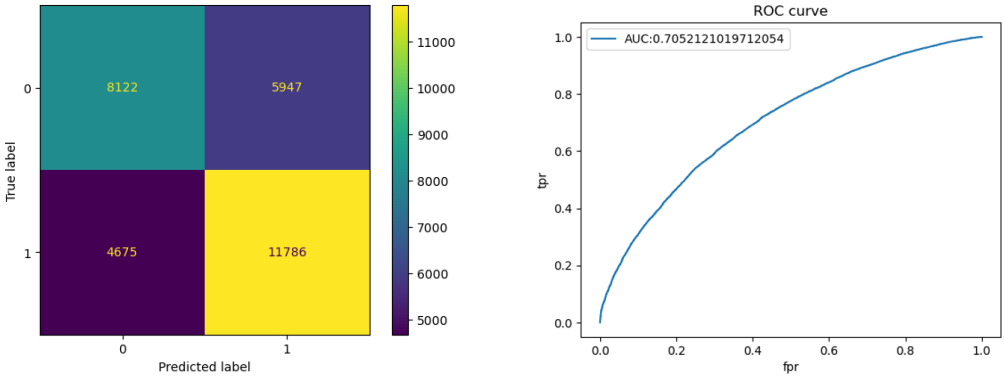
(b) Executing command **start-dfs.sh**



(c) Executing command **hadoop start-all.sh**

(d) Executing command **spark-submit –total-executor-cores 3 –master spark://172.31.58.70:7077 model.py**

Figure 3: Snapshot of each command execution



(a) Confusion matrix

(b) ROC curve with AUC:0.71

Figure 4: Evaluation results of the model

"Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1). AUC provides an aggregate measure of performance across all possible classification thresholds. AUC can be interpreted as the probability that the model ranks a random positive example more highly than a random negative example. In our case, the AUC of the model using all the features is 0.71. Fig. 5 shows the importance of each feature in descending order. Among all the features, the number of lab procedures, diagnosis 1, diagnosis 2, diagnosis 3, and the number of medications, show a more dominant influence on the final prediction. We also compared the performance degradation after using fewer features. We retrained a model only using the first 10 most important features. The AUC of the simplified model is 0.6. This observation inspired us that we can make a tradeoff between the model size and the accuracy.

Fig. 6 shows our user interface. Users fill in their own parameters and then click the button “Predict”. The prediction will appear on a new page.

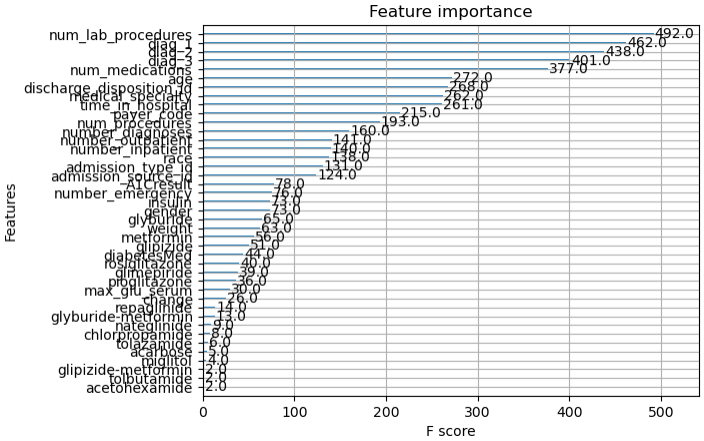


Figure 5: F score of each feature after PCA

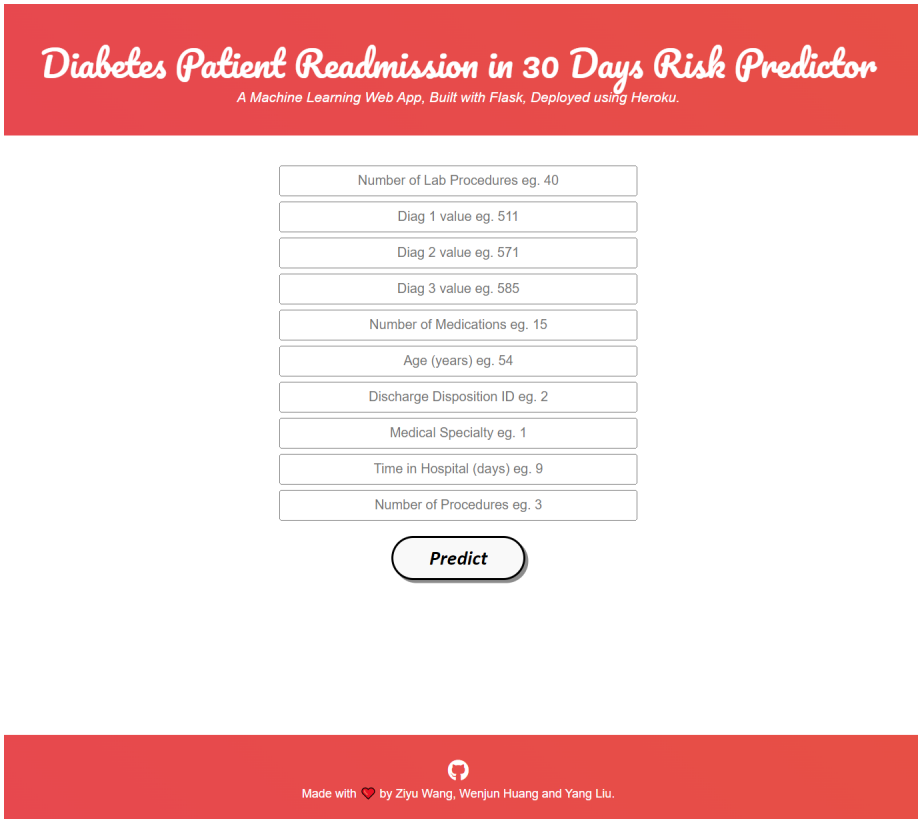


Figure 6: User interface

6 Conclusion

Our project focused on developing a production-ready distributed training framework for smart healthcare applications that is computationally efficient and resource-friendly. One of the key applications of our framework was the development of a diabetes risk prediction system, which accurately predicts a patient's risk of readmission to the hospital based on their medical history and other relevant factors. Our distributed training approach achieved these results quickly and efficiently, without placing undue strain on any individual machine. Overall, our framework represents a significant step forward in the development of smart healthcare applications, providing a more efficient and scalable approach to training machine learning models that can empower individuals and healthcare professionals to make better-informed decisions and improve patient outcomes.

7 Future Work

Naturally, healthcare applications encounter issues regarding privacy. In this study, we presume that the centralized data management clusters are reliable and will not divulge patients' privacy to external parties. We acknowledge that privacy concerns require further investigation, and suggest that federated learning, a form of distributed training, may be a viable solution. Additionally, we propose the utilization of differential privacy to ensure secure model aggregation on the centralized clusters.

References

[1] Dhruba Borthakur. The hadoop distributed file system: Architecture and design. Hadoop Project Website, 11(2007):21, 2007.

[2] Beata Strack, Jonathan P DeShazo, Chris Gennings, Juan L Olmo, Sebastian Ventura, Krzysztof J Cios, and John N Clore. Impact of hba1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records. BioMed research international, 2014, 2014.

[3] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. Apache Spark: A unified engine for big data processing. Communications of the ACM, 59(11):56–65, nov 2016.

[4] Mark H Zweig and Gregory Campbell. Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine. Clinical chemistry, 39(4):561–577, 1993.