

PREDICTING THE MATERNAL RISK DURING CHILDBIRTH

Hao Lan

February 28, 2022

CMP-SCI 5300
Semester project

1 Introduction

The maternal risk is determined by several factors during the childbirth, this project is aim to classify risk levels and predict whether the maternal is in danger. Even though there are many aspects related to the process of childbirth, the goal here is to build a neural network that distinguish high/low risk of maternal death regarding to the given data categories.

In the selected data set "Maternal health risk" [1], there are total of 1014 instances with no missing values. The attribute age, systolic/diastolic blood pressure, blood sugar, body temperature, and heart rate are considered given inputs, which are all close related to the status of maternal during the birth and by defining what factors lead to a high risk, corresponding prevention may applied in order to keep maternal away from death.

Personally, making model according to problem that related to health care is more meaningful compare to any other classification. Even though this is a quiet brief model of neural network, which might not ever get chance to become a real-world application, the given input seem very reasonable and relevant in order to determine the output of risk level, which gives more credibility in the prediction process.

2 Data Set

2.1 Data cleaning

Reform the multi-classified data into binary, as known as data cleaning for selected data set is straight forward, age, systolic/diastolic blood pressure, blood sugar, body temperature, and heart rate are all integer values and they are not required to be binary since these are going to be our inputs, the output value, risk-level, is 3-variable classification defined by high/mid/low risks. where $high(272) < mid(336) < low(406)$. Considering the balance of the data set, we merge mid-risk class into high-risk category, which are denoted by '1' while low risk is denoted by '0'. By calculating with the formula:

$$x_h = \frac{x_{high}}{sum(x)}, x_l = \frac{x_{low}}{sum(x)} \quad (1)$$

We have results:

level	Number	Percentage
High(1)	608	59.96%
Low(0)	406	40.04%
Total	1014	100%

Table 1: Two classes are distributed in balance

2.2 Data Normalization

To achieve feature scaling, because we have no extreme value among the inputs, standardization is not necessary, the function applied is the mean-range normalization function[2]:

$$x' = \frac{X - \mu}{X_{max} - X_{min}} \quad (2)$$

Usually, feature scaling is used when each category of data do not have the same range scales, which in our case, the value range of systolic blood pressure obviously differs from blood sugar; these difference can slow down the learning of a model. When we apply Gradient Descent in both normalized and non-normalized data set, Gradient Descent converges to the minimum faster if the input is normalized.

Now, let's give a taste of mean-range normalization, the charts shown below indicate the first 3 rows of data before and after normalization:

Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate
25	130	80	15	98	86
35	140	90	13	98	70
29	90	70	8	100	80

Table 2: First 3 rows before normalization

Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate
-0.08	0.19	0.07	0.48	-0.13	0.14
0.09	0.30	0.27	0.33	-0.13	-0.05
-0.01	-0.26	-0.13	-0.06	0.27	0.07

Table 3: First 3 rows after normalization

As the tables show, the range of each attribute is narrowed down to same interval $[-1, 1]$. The attribute In-risk(RiskLevel before cleaning), is exclusive.

2.3 Data Overview

Before constructing the model, It is better to go through the data set in detail.

2.3.1 Histogram

We can see the distribution via histogram, the graph shown below will illustrate distribution of each attribute, BloodSugar, BodyTemp, and HeartRate are reasonably centralized:

Figure 1)

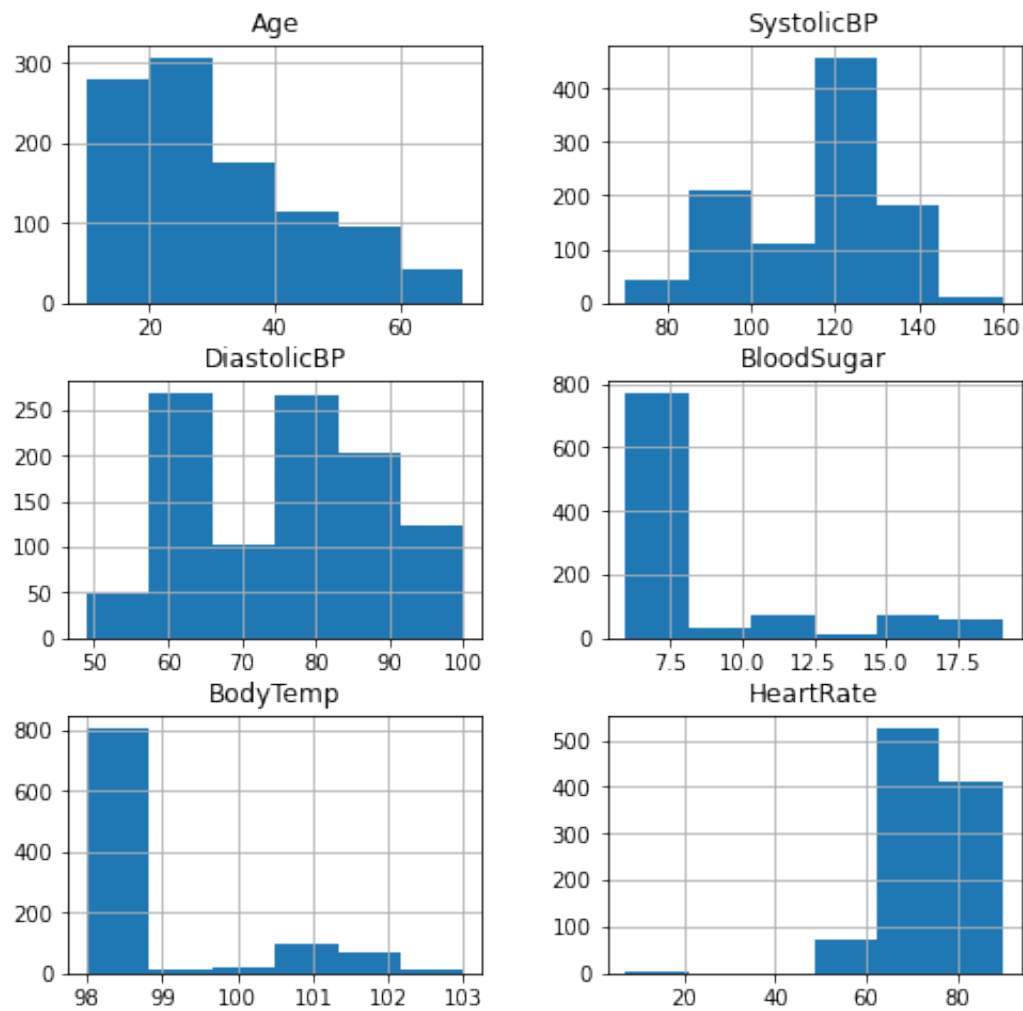


Figure 1: the distribution histogram of each input attribute

Figure 2)

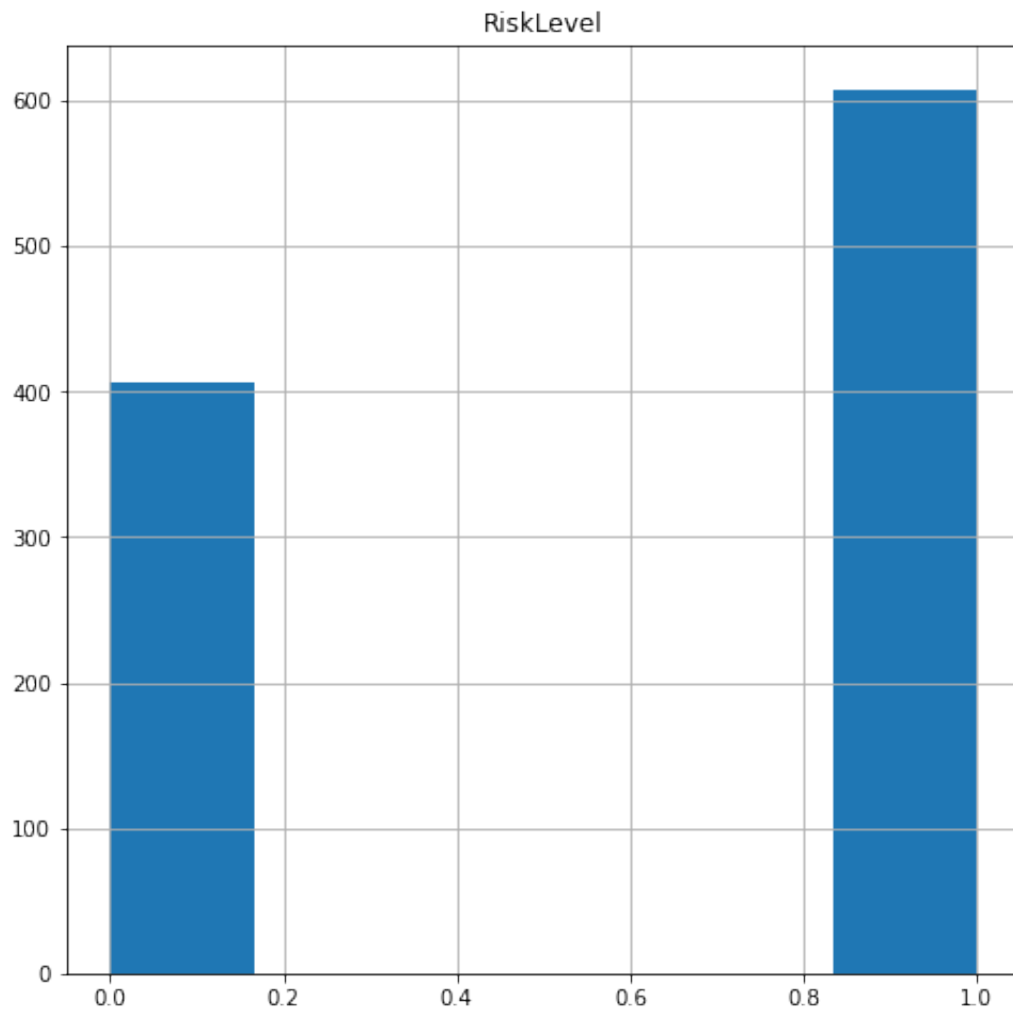


Figure 2: the distribution histogram for output attribute

2.3.2 data significance

Age: The maternal sample's age, extreme age is more likely to develop pregnancy-related high blood pressure and anemia (lack of healthy red blood cells).

Systolic Blood pressure: indicates how much pressure your blood is exerting against your artery walls when the heart beats. Monitoring blood pressure is important before, during, and after pregnancy.

Diastolic blood pressure: The pressure of maternal sample's blood exerting against artery walls while the heart is resting between beats. Monitoring blood pressure is important before, during, and after pregnancy.

Blood sugar: blood sugar concentration, High blood glucose can increase the chance that maternal samples will have a miscarriage or damage of the health

Body temperature: changes of body temperature can influence metabolic changes in different nutrients of pregnant women.

Heart rate: During childbirth, maternal sample's heart pumps more blood each minute and heart rate increases, especially when push, they will have abrupt changes in blood flow and pressure.

Besides, all other related information are contained in the table shown below:

	max	min	mean	median	std
age	70.00	10.00	29.87	26.00	13.47
systolicBP	160.00	70.00	113.20	120.00	18.39
diastolicBP	100.00	49.00	76.46	80.00	13.88
BloodSugar	19.00	6.00	8.73	7.50	3.29
BodyTemp	103.00	98.00	98.67	98.00	1.37
HeartRate	90.00	7.00	74.30	76.00	8.08

Table 4: Input data overview information

3 Modeling

In this section, I will show how our data set fits into different modeling mechanisms and how each one of the techniques affects our accuracy. In order to do this, we apply python library called Tensorflow.

1. `from tensorflow.keras.models import Sequential`
2. `from tensorflow.keras.layers import Dense`

3.1 logistic regression

First of all let's see how does the data set fit the logistic regression model. After assigning variable `model = Sequential()`, we now have a deep learning modeling environment where we can add layers into the network, every unit in a layer is connected to every unit in the previous layer. However, for now we will have only one layer with one node to do the logistic regression[3].

In order to compile the model, we use following function parameters:

`<loss = 'binary_crossentropy'>`

Since this is binary classification model (0 or 1), we use it to determine our loss.

`<optimizer = 'rmsprop'>`

'rmsprop' is a gradient based optimization technique, it uses an adaptive learning rate which changes over time.

`<metrics = 'accuracy'>`

Which represents the overall Accuracy that is essentially tells us out of all of the reference sites what proportion were mapped correctly.

After setting epoch to 256 and repeat several times, here is result of last 5 training of our model in last reputation of epochs of 256:

epochs	loss	accuracy
252/256	0.5580	0.6726
253/256	0.5516	0.6746
254/256	0.5556	0.6785
255/256	0.5646	0.6686
256/256	0.5564	0.6716

Table 5: single layer training result

Below is the first 10 predictions the model makes compare to original output 'Y' value (bad predictions are highlighted):

TrueValue	1.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
Prediction	0.99	0.95	0.26	0.52	0.40	0.59	0.61	0.63	0.26	1.00

Table 6: single layer accuracy comparison

Since we are using only one layer here, I think this is a acceptable result, however, to achieve higher accuracy, we might build a model with multiple layers and nodes.

3.2 multi-layered model

A **multi-layered** model is more precise than what we have seen in 3.1 since it involves more weights/rates and biases that tweak the prediction over and over.

In this model, I have added up to 4 layers in total, where first layer has 16 node, and second layer has 8 nodes, 4 for third layer, and also, last node in fourth layer. The thing we should pay attention here is that, first three layer we will use **activation = 'relu'** since they are not binary classification, and last layer we use **activation = 'sigmoid'**. Below is the description chart of layers:

Figure 3)

Model: "sequential_5"

Layer (type)	Output Shape	Param #
dense_18 (Dense)	(None, 16)	112
dense_19 (Dense)	(None, 8)	136
dense_20 (Dense)	(None, 4)	36
dense_21 (Dense)	(None, 1)	5
Total params: 289		
Trainable params: 289		
Non-trainable params: 0		

Figure 3: Neural networks in layers

As we can see, there are 289 params since we have 6 categories of input plus bias, which is $(6 + 1) * 16 + (16 + 1) * 8 + (8 + 1) * 4 + (4 + 1) * 1 = 289$, another good way to see this is via visualization (see **Figure 4**).

Just like 3.1, below is the table showing last 5 execution of running epochs, this time I set epochs = 2560 and 4 repeated 4 times.

epochs	loss	accuracy
2556/2560	0.1928	0.9043
2557/2560	0.2018	0.9093
2558/2560	0.2034	0.9083
2559/2560	0.2025	0.9043
2560/2560	0.1963	0.9132

Table 7: multi-layered training result

Figure 4)

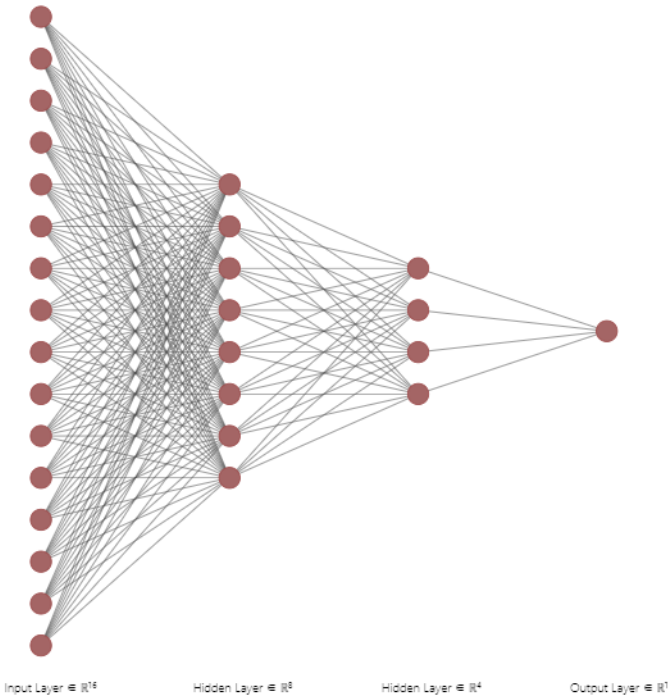


Figure 4: multi-layered Neural networks model

To see how well it can predict after all, here are first 10 values of comparison between true values and predictions where bad predictions are highlighted:

TrueValue	1.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
Prediction	1.00	1.00	1.00	1.00	0.48	1.00	1.00	1.00	0.43	1.00

Table 8: multi-layered accuracy comparison

It is obvious that multi-layered model performs better than our single-layered model, there are 8 perfect guess in first 10 values.

References

- [1] Marzia Ahmed. UCI maternal health risk data set data set, 2020.
- [2] K. Adith Narasimhan. Mean normalization and feature scaling - a simple explanation. <https://medium.com/analytics-vidhya/mean-normalization-and-feature-scaling-a-simple-explanation-3b9be7bfd3e8#:~:text=Mean>, 2021. Accessed: 2021-2-11.
- [3] fchollet. The sequential model. https://keras.io/guides/sequential_model/, 2020.