| Team Number : | 0921 |
|---|---|

| Problem Chosen : | A |
|---|---|

2017 APMCM summary sheet

The diagnosis of diseases related to sleep is of profound significance to enhance people's sleep quality, which plays a crucial role in our health. The core objective of this paper is to establish a mathematical model to describe the relationship among sleep conditions, sleep qualities and possible related diseases, giving patients diagnoses and advice to recover from diseases by their sleep conditions.

In the first question, we visualize the relationship between the indicators and the sleep quality, classify the indicators by their score value and analyze the correlation between them with variance analysis. Also, we judge the relationship by the variance and determine indicators as unrelated if its corresponding variance is lower than a certain threshold value.

In the second question, we clean the given data and construct a novel neural network to classify patients with different type of diagnoses based on their sleep conditions. The networks, namely Reclassified Deep Neural Networks(RDNN), with two layers of classifier integrated in, solve the data skew problem and behave well with accuracy up to 89% on test dataset.

In the third question, we feed the given data into RDNN and output the result as the predicated classified diagnoses, and analyze the probability of the correctness of our diagnoses.

In the fourth question, we design a sleep program to arrange the rest time for people. Questionnaire is designed to collect information of the sleep quality as the sleep indicators. Several influence factors of certain people, such as the bedtime routine, are taken into consideration. RDNN enables us to give people advice on the way they should arrange their rest time.

**Keywords:** variance analysis; multitask classifier; reclassified deep neural networks

# Contents

# 1. Introduction

## 1.1 Background

It goes without saying that all human beings need to sleep every day. Sleeping is indispensable in our daily life. A low sleep quality or the lack of sleeping may harm our body to a large extent, and may bring us various kinds of diseases. Therefore, we construct a mathematical model to help people enhance their sleep quality and warn people about the potential danger of sleep-related diseases. We then begin our discussion about the sleeping and the diseases.

## 1.2 Restatement of the Problem

Given the current situation, we should appeal to people to pay special attention to bed rest to ensure a good and healthy body. Therefore, it is of great significance to find the key indicators that affect people's sleep quality and give accurate diagnosis to the patients with sleep-related diseases based on their sleep conditions. What's more, helping people arrange their rest time for better sleep and health is also included.

To this end, we need to solve the following four problems:

(1) Analyze the relationship between the indicators given and the quality of sleep according to the data in Annex I, if there is no correlation between one or several indicators and sleep quality, find it or them out and delete.

(2) Analyze the relationship between the diagnosis results and sleep.

(3) Assuming we are doctors, make diagnosis to the patient based on the data in Annex III, and give our diagnosis result.

(4) Develop appropriate sleep program to scientifically arrange our rest time for health of the body, and evaluate its effectiveness.

# 2. Problem Analysis

## 2.1 Analysis of Problem 1

We visualize the relationship between the indicators and sleep quality by various kind of graphs including histograms, line graphs and thermodynamic charts (namely heat maps). These graphs enable us to better analyze the correlation between them, providing us clear information about the general structure and distribution of the data which accelerate the drafting of our plan. We carefully separate indicators and do research on them individually by calculating the variance of sleep qualities under several on-score-classified indicators. Generally speaking, we judge the relationship by the variance and determine an indicator as unrelated if its corresponding variance is lower than a certain threshold value.

### 2.2 Analysis of Problem 2

Given the situation that there are mass data of the patients' diagnosis and sleep indicators, between which sophisticated relationships exist, it is essential to construct a complete mathematical model to fit the given data, predicting the new patients' physical condition and giving them accurate diagnosis. Therefore, **Reclassified Deep Neural Networks (RDNN)** is constructed, which will be illustrated later.

### 2.3 Analysis of Problem 3

Once the mathematical model in problem 2 is constructed, all we need to do is to feed the RDNN with the scores of patients' sleep condition, and output the diagnosis results from the networks. Probability for each disease will be given correspondingly for any patient according to their sleep condition.

### 2.4 Analysis of Problem 4

In order to develop a scientific and reasonable sleep program, we read the articles for rest and sleep from the website of National Sleep Foundation (NSF) and design a questionnaire to collect information of the sleep quality indicators. Using the indicators as input for RDNN mentioned above, we get the possible diseases that the certain interviewee may have. This information enables us to give people advice on the way they sleep and better arrange their rest time.

## 3. The Model Assumptions

To simplify the question and make it convenient for our model in training and testing, we make the following basic assumptions, each of which is properly justified.

The distribution of diagnosis is related to the real condition. In annexes, some diagnoses are common among thousands of patients while others are shared only by several people. This preliminary result can be a basis for dividing 122 kinds of diagnoses into 8 groups (7 groups are common diseases and 1 group is uncommon).

All cases that used to train the model are patients. We consider that all data in annexes are from real patients and they are reliable.

The indicators in Annex I.xlsx are independent of each other.

We ignore the impact of physical diseases of human in data.

## 4. Symbol Description

| Symbols | Explanation |
|---|---|
| $\mathbf{RDNN}$ | Reclassified deep neural network |
| $\boldsymbol{R(x)}$ | Supervised label |
| $\boldsymbol{f}[\cdot]$ | The process of Reclassified deep neural network |
| $\boldsymbol{I^0(x)}$ | Input data |
| $\boldsymbol{L(R,O)}$ | The objective function of model |
| $\boldsymbol{N}$ | The number of training dataset |
| $\boldsymbol{n1,2,3}$ | The number of hidden layer 1, 2, 3 |
| $\boldsymbol{\sigma_1\sigma_2}$ | The activation function |
| $\boldsymbol{I^{1,2,3}}$ | The layers of RDNN network |
| $\boldsymbol{W^{1,2,3}}$ | The weights of RDNN network |
| $\boldsymbol{b^{1,2,3}}$ | The bias of RDNN network |
| $\boldsymbol{Relu}$ | Rectified linear units |
| $\boldsymbol{SVM}$ | Support vector machine |
| $\boldsymbol{Eff}$ | The efficiency of sleep |
| $\boldsymbol{Score(x)}$ | Score of indicator x |

## 5. Modeling Establishment and Solution for Problem 1

We draw a score-ratio line chart for each possible indicator group based on the patients' sleep quality. These charts are given from figure 5.1 to figure 5.4 below. In

each chart, patients with different sleep quality from 0 to 3 are divided correspondingly to group0, group1, group2 and group3. (The indicators age and sex will be illustrated later.)

A classification is done: for each indicator, we divide all the patients into 10 parts as 1 to 10, each indicating 10 points of score range of the certain indicator, for example, 1 means [0,10], 2 means [10,20], etc., which singularizes the relationship between them. These charts are given from Figure 5.1 to Figure 5.5 below.

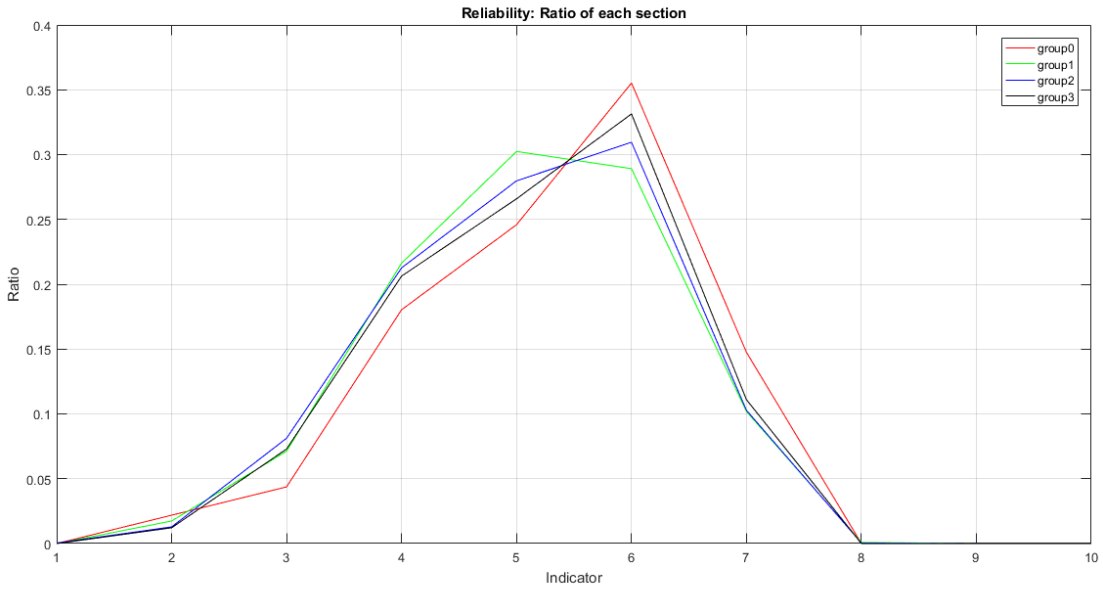


Figure 5.1 After classification: Patient ratio of reliability score group by sleep quality

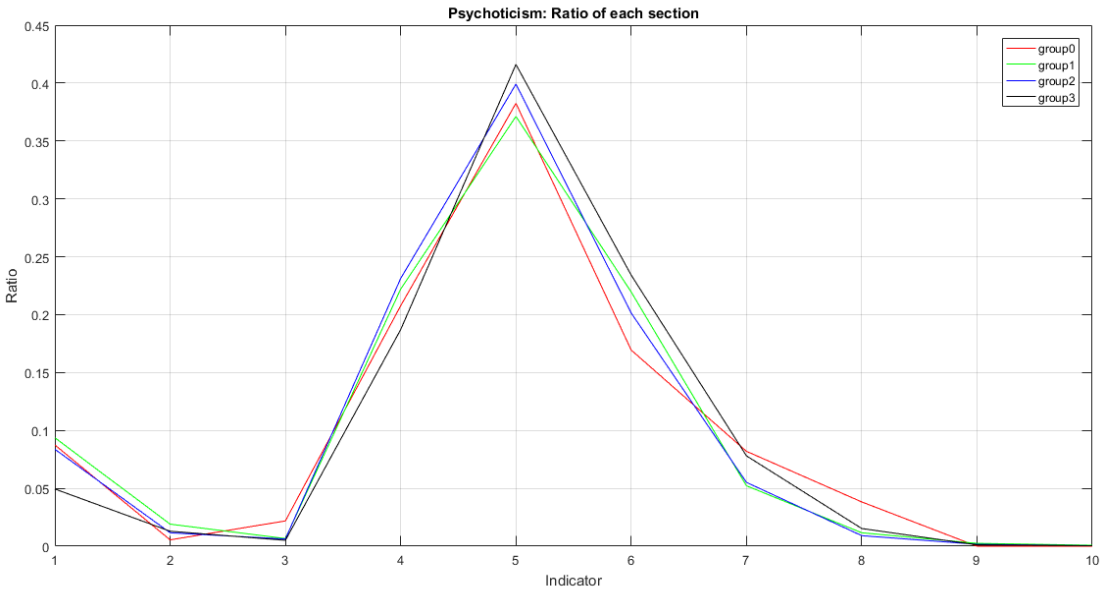

Figure 5.2 After classification: Patient ratio of psychoticism score group by sleep quality

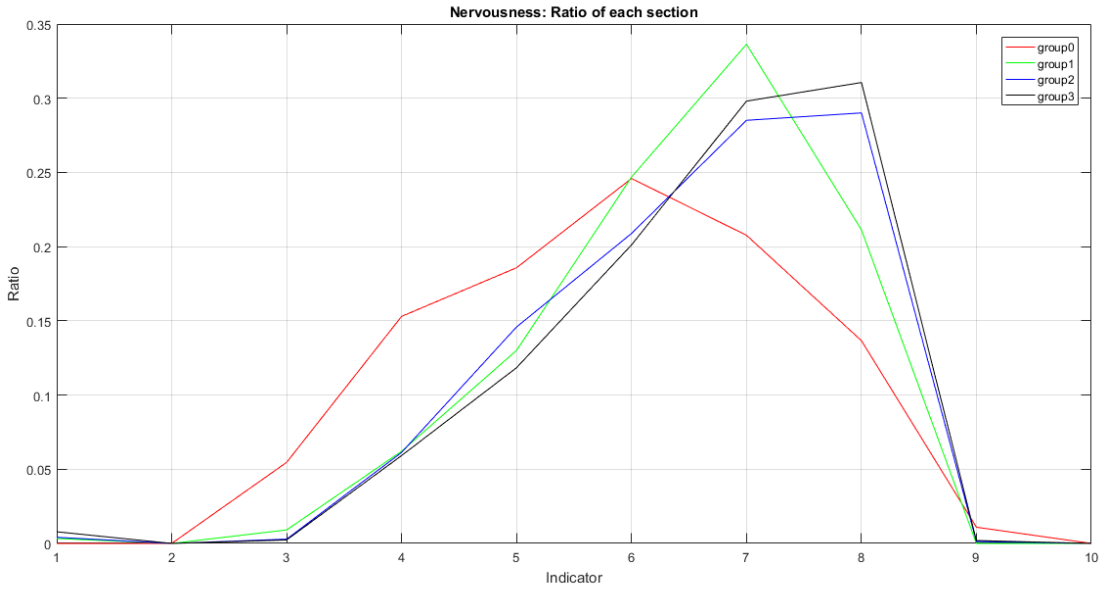


Figure 5.3 After classification: Patient ratio of nervousness score group by sleep quality

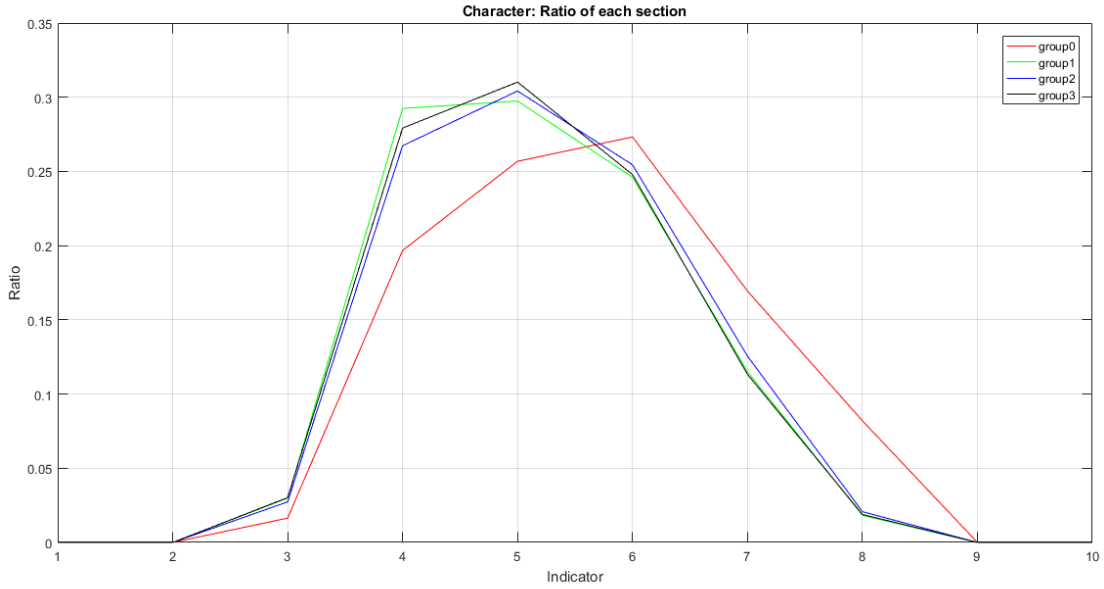


Figure 5.4 After classification: Patient ratio of character score group by sleep quality

From the above figures, the relationship becomes singularized, but there is no crucial criterion to figure out which indicators are unrelated to people's sleep quality. We therefore draw a histogram comparing the variance of classified sections by the score of each indicator in figure 5.5. We define a **relationship** as follow: if the variance among data is lower than 10, the relationship is weak and the indicator can be neglected.

According to this principle, **indicator Reliability and Psychoticism should be deleted**. (Our data are enlarged by 10 times)

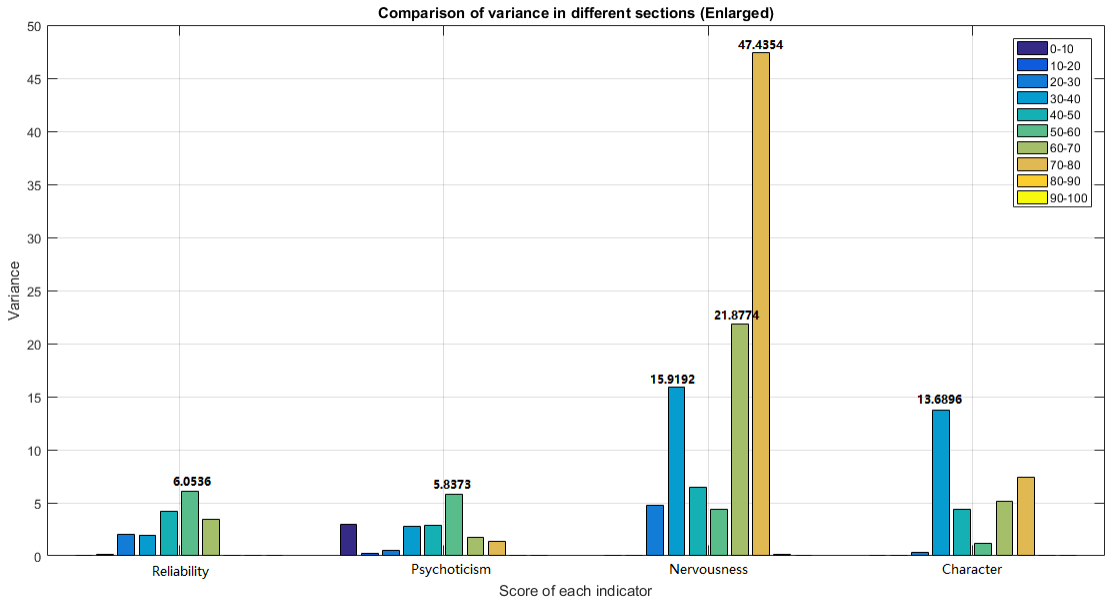


Figure 5.5 Comparison of variance in different sections of the score of each indicator

Following are the thermodynamic diagrams showing the relationship between the classified indicators score and people's sleep quality, which singularize the result and the correctness of our conclusion.

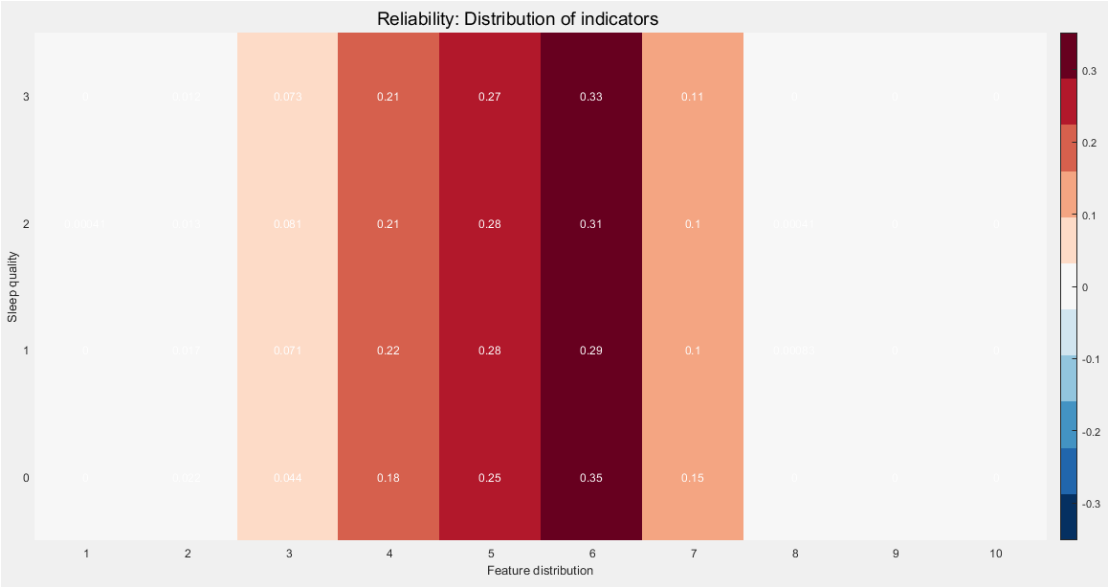


Figure 5.6 The relationship between the classified reliability score and sleep quality

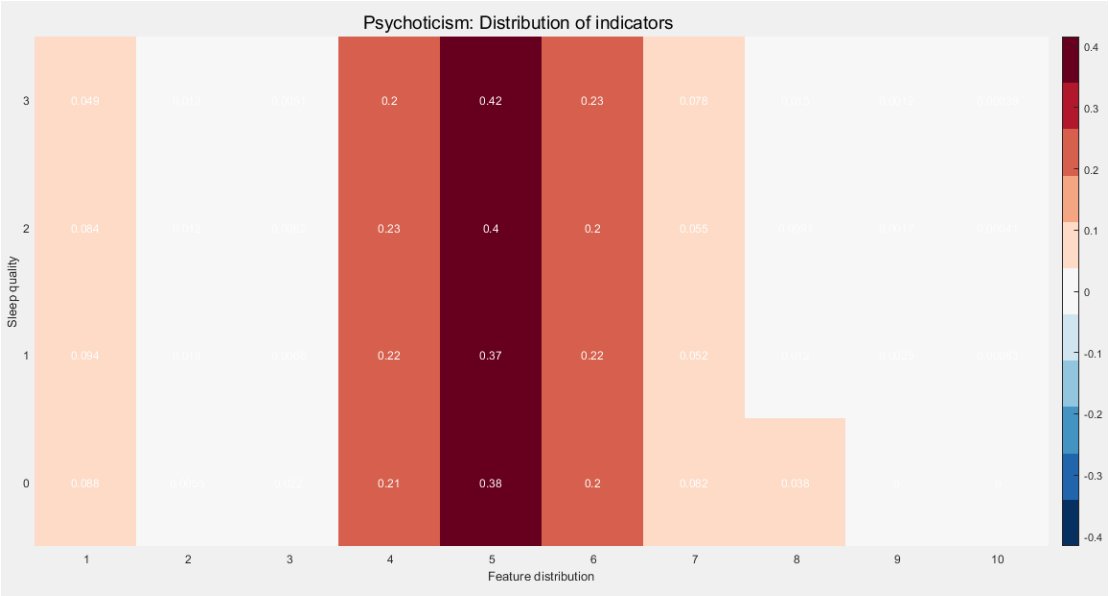


Figure 5.7 The relationship between the classified psychoticism score and sleep quality

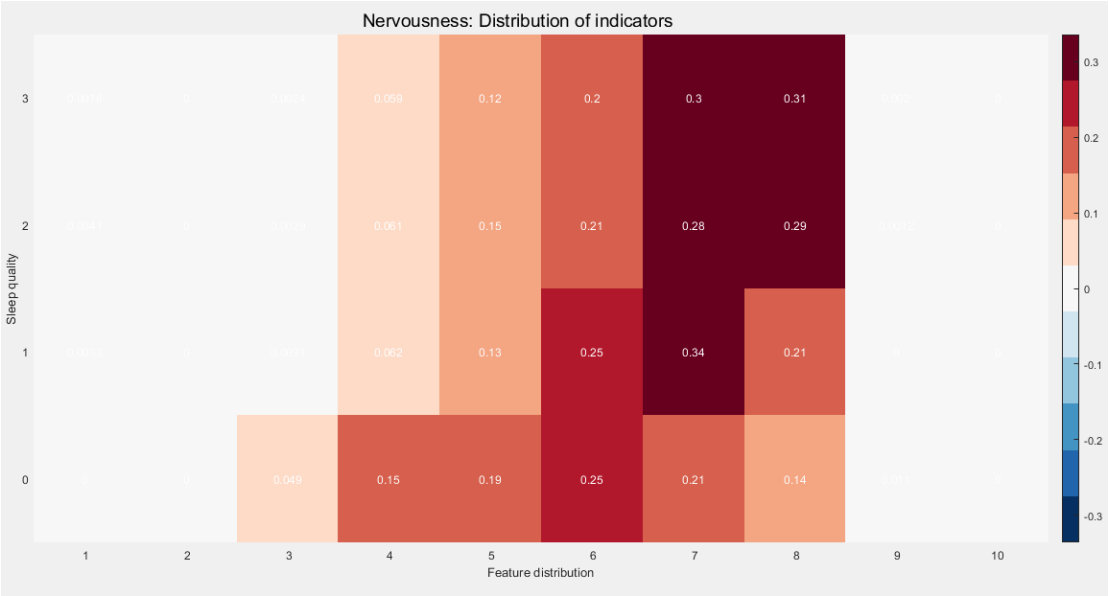


Figure 5.8 The relationship between the classified nervousness score and sleep quality

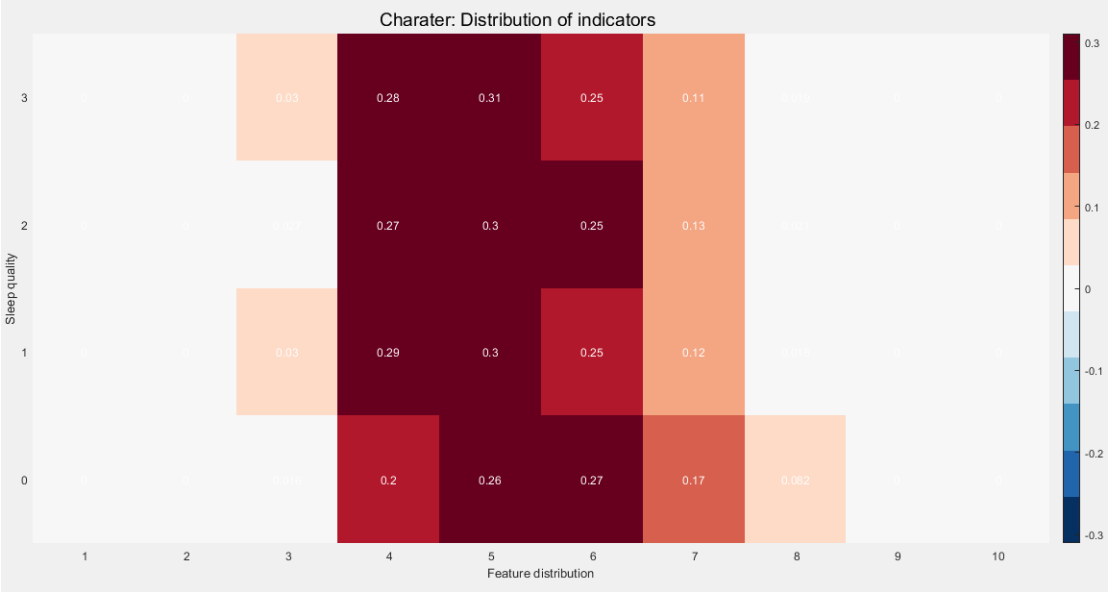


Figure 5.9 The relationship between the classified character score and sleep quality

What's also worth mentioning is the age and sex, their variance and ratio histograms are as follows. From what we have proposed above, **the indicator sex is unrelated to sleep quality and should be deleted**.

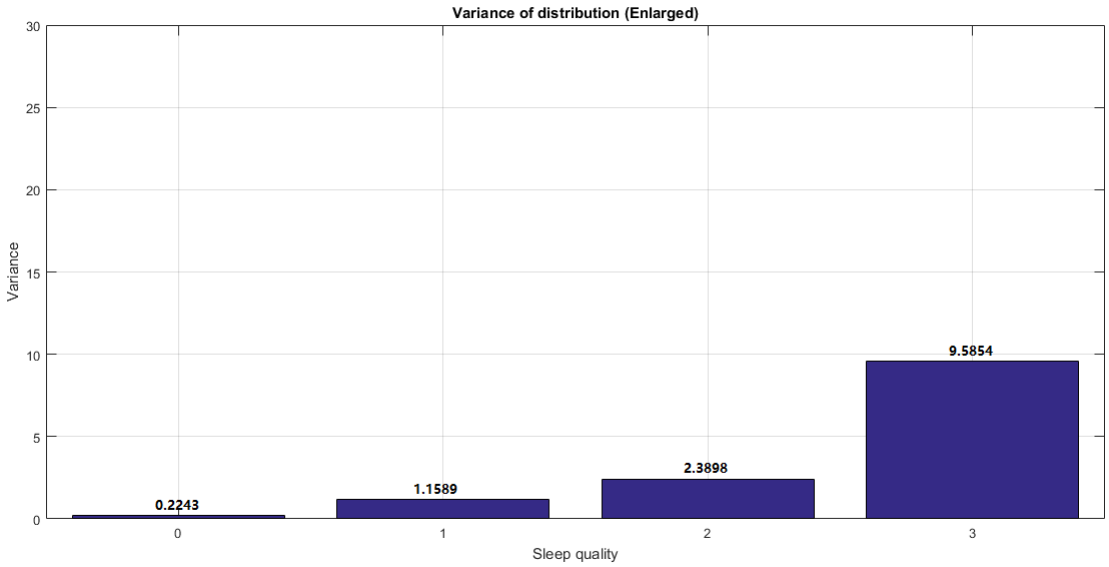


Figure 5.10 The distribution of sex's variance

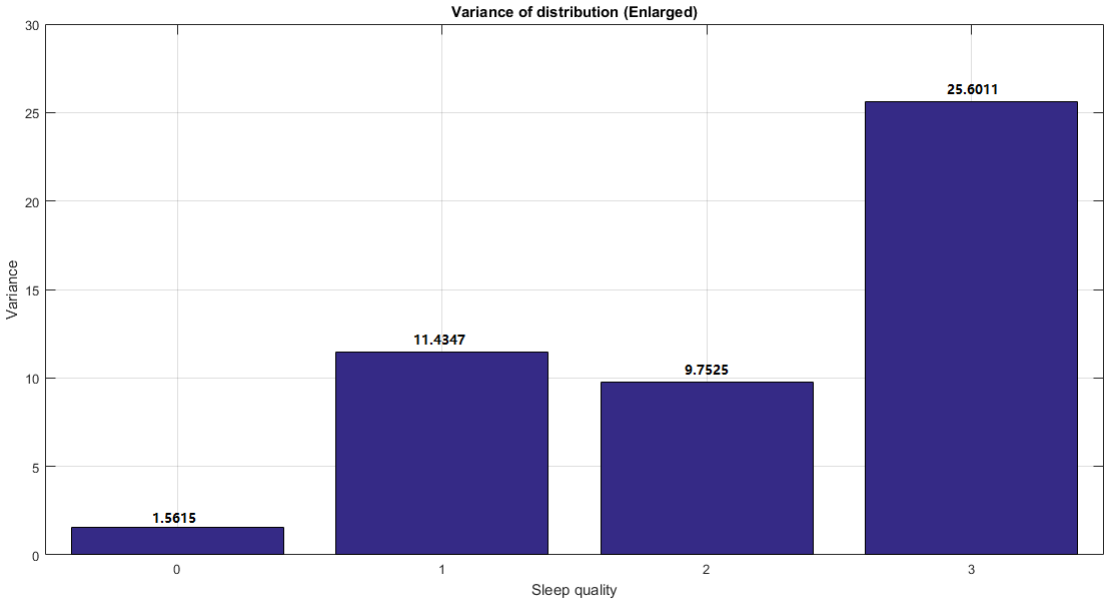


Figure 5.11 The distribution of age's variance

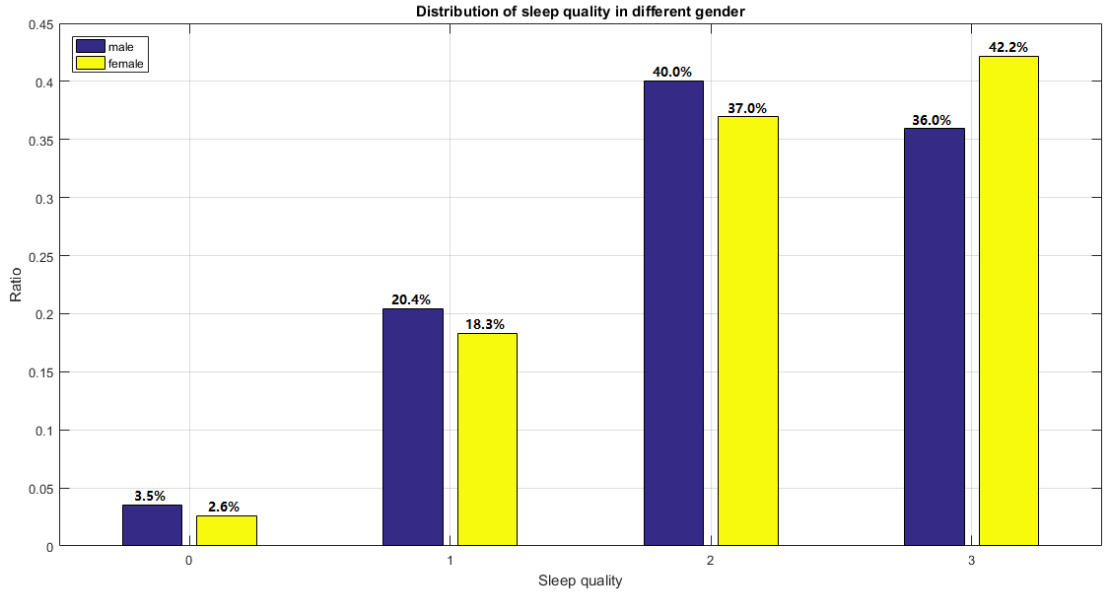


Figure 5.12 The distribution of sleep quality in different gender

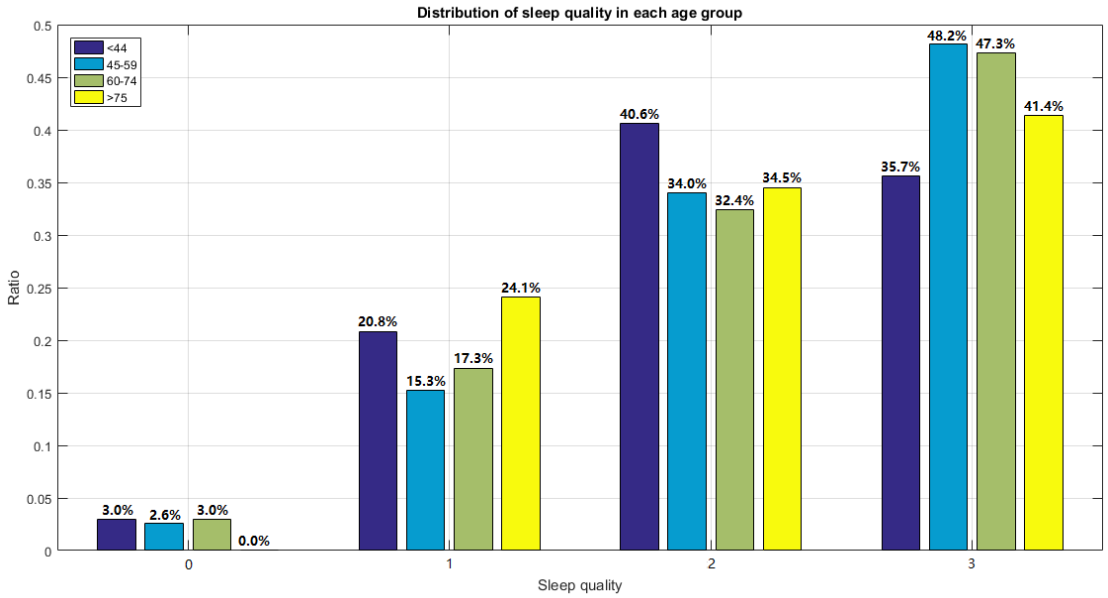


Figure 5.13 The distribution of sleep quality in different age

Conclusion: **indicator sex, reliability and psychoticism should be deleted.**

## 6. Model Establishment and Solution for Problem 2 and 3

### 6.1 Reclassified Deep Neural Networks (RDNN)

We propose a new multitask classifier model. The input layers are the patients' score of seven indicators, such as [0, 1, 3, 2, 0, 0, 1], numbers in [] indicate the score of patients' sleep situation. A classifier model is designed to classify our diagnosis through indicators, which minimizes the objective function as follows:

$$L(R, O) = -\sum_{n=1}^{N} R(x) * log\{f[I^0(x)]\} \quad (1)$$

Where N is the number of training dataset and n indexes the sample, $f[\cdot]$ is our classifier model. $I^0(x)$ is the input layer obtained from Annex II.xlsx. $R(x)$ is the supervised label. Our multitask classifier architecture is shown in Figure 6.1.

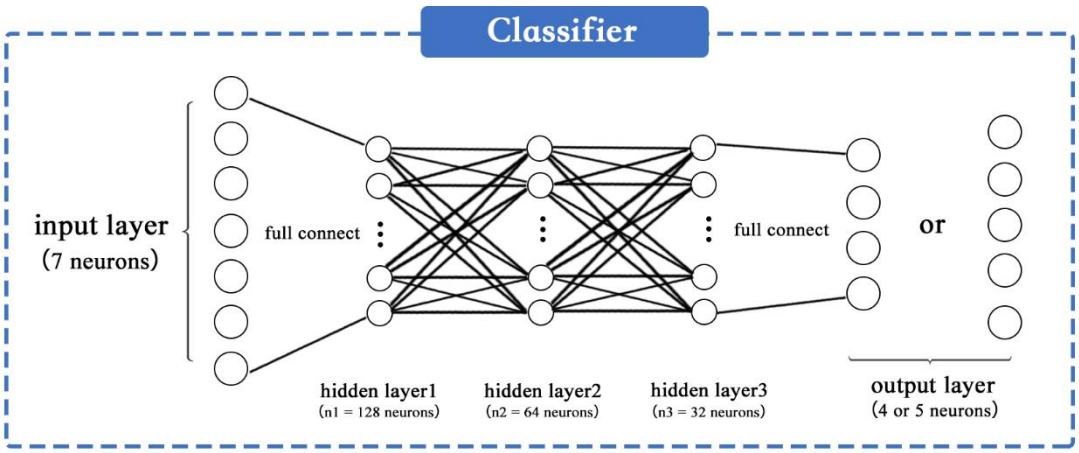


Figure 6.1 The proposed classifier architecture for multi-classification task

Moreover, our classifier consists of three full-connected layers (hidden layers) and one output layer, which operates according to the rules as follow:

$$I^0 = input\ layer \tag{2}$$

$$I^l = \sigma_1(W^l * I^{l-1} + b^l) \qquad l = 1,\ 2,\ 3 \tag{3}$$

$$0 = \sigma_2(W^l * I^{l-1} + b^l) \qquad l = 4 \tag{4}$$

where $l$ =1, 2, 3 indexes the layer number, "*" denotes point-wise multiplication operator, $W^l$ is trained weight and $b^l$is bias. The $\sigma_1$ is ReLU nonlinear activation function [1]. The ReLU is effective to truncate the parameters less than 0, which makes the classifier network parameters take normal values in three hidden layers. The $\sigma_2$ is softmax activation function, which is used to classify and observe the output layers intuitionally.

For the first layer in classifier, we use filters of size 7*128 to obtain 128 feature maps by 7 input channels. For the second layer, the size of filters is 128*64, which can refine our classifier. For the third layer, the size of filters is 64*32, which makes our classifier more exquisite. For the last layer, the size of filters is 32*4 or 32*5, which turns 32 feature maps into 4 or 5-dimension output result.

As the given data in Annex II skews, we establish a novel structure named Reclassified Deep Neural Networks (RDNN) to train our data. RDNN follows the rule of the above-mentioned classifier, and integrates two classifiers by placing them in different layers with the output of layer I as the input of layer II. This integration can be used as a multi-label precision classification. The architecture of RDNN is shown in the figure 6.2.

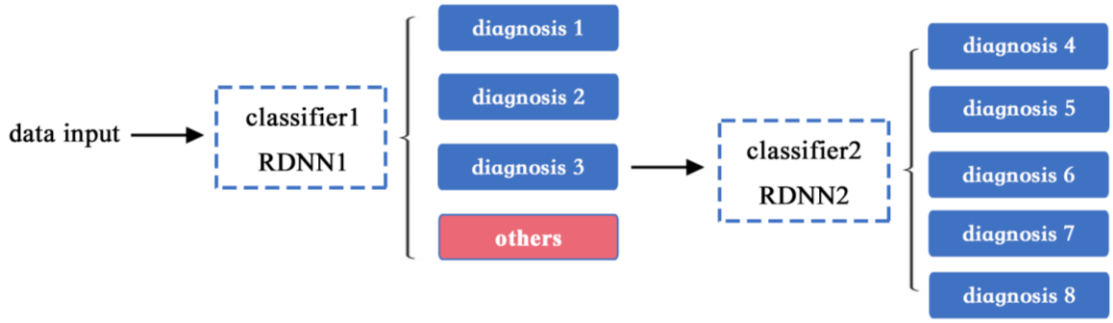


Figure 6.2 The proposed Reclassified Deep Neural Networks(RDNN) architecture

Different from full-connected neural networks, RDNN applies a reclassification in data preprocessing procedure to solve the data skew problem, which effectively enhances the output result produced by the networks. To balance the data amount in different classification, RDNN involves multiple layers, whose individual members are correspondingly defined as the first-class members (in the first layer), second-class members (in the second layer), and so on.

RDNN integrates two classifiers by placing them in different layers. Expected output results of the networks, or in other words various classifications of the patients, should be generalized into less category and be marked as different class of members in different layers in order to keep the uniform distribution of classified data, which will be illustrated later.

## 6.2 Preprocessing

To solve the problem 2, we initially clean all the input data, removing the problematic and abnormal diagnoses which contain "?" or diseases not covered in *World Health Organization(WHO) Diseases List* like "Xia Yong". During this process, any row containing blank cell or invalid data in the given data table is removed. Only 122 kinds of diagnoses remain to be taken into consideration. In order to quantify the diagnoses, we reshape the data and number all diagnoses from 1 to 122 accordingly.

In order to better analyze the diagnoses using RDNN, we classify them into several groups. The diseases and/or diagnoses with occurrence rate higher than 0.5% is listed below in table 6.1. For accurate analysis, the 7 diagnoses which occur more than 100 times will be respectively the first-class members (57, 33, 6, 5, 74, 15, 56) in classification, and all the other diagnoses with low occurrence rate will be classified together as only one first-class member. At the same time, in order to keep the uniform

distribution of classified data, we re-mark several first-class members (6, 5, 74, 15, 56) to second-class ones to move them to the second layer of RDNN. In order to connect the output of layer I and the input of layer II, these second-class members are marked 3 in first layer. Therefore, we have the new number of diagnosis classification in table 6.2 and 6.3.

| No. | Diagnosis | Occur Times | No. | Diagnosis | Occur Times |
|---|---|---|---|---|---|
| 57 | Sleep disorder | 1703 | 56 | Non-Organic Insomnia | 106 |
| 33 | Depression | 1472 | 42 | Recurrent Depressive Disorder | 74 |
| 6 | Anxiety disorder | 865 | 28 | Adjustment Disorder | 58 |
| 5 | Anxiety | 403 | 1 | Schizophrenia | 47 |
| 74 | Mixed Anxiety And Depression | 368 | 115 | Consultation | 37 |
| 15 | Bipolar Affective Disorder | 132 | 43 | Mixed Anxiety And Depression Disorder | 32 |

Table 6.1 The diagnoses with occurrence rate higher than 0.5%

| First-class Member No. | Original Diagnosis No. | Diagnosis |
|---|---|---|
| 1 | 57 | Sleep disorder |
| 2 | 33 | Depression |
| 3 | 6, 5, 74, 15, 56 | (Multiple Diagnoses) |
| 4 | (Multiple Numbers) | Others |

Table 6.2 The first-class members in RDNN

| Second-class Member No. | Original Diagnosis No. | Diagnosis |
|---|---|---|
| 1 | 6 | Anxiety disorder |
| 2 | 5 | Anxiety |
| 3 | 74 | Mixed Anxiety and Depression |
| 4 | 15 | Bipolar Affective Disorder |
| 5 | 56 | Non-Organic Insomnia |

Table 6.3 The second-class members in RDNN

After the reclassification and before we feed the RDNN with the scores for patients'

sleep condition as input, we carefully analyze the correlation between the diagnosis and sleep conditions by drawing histograms of the distribution of patients' single sleep condition score and its ratio grouped by various diagnoses. The definitions of 8 types of diagnoses are given in table 6.4. The histograms as analysis results are given in figure 6.3.

| Types | Original Diagnosis No. | Diagnosis |
|---|---|---|
| type1 | 57 | Sleep disorder |
| type2 | 33 | Depression |
| type3 | 6 | Anxiety disorder |
| type4 | 5 | Anxiety |
| type5 | 74 | Mixed Anxiety And Depression |
| type6 | 15 | Bipolar Affective Disorder |
| type7 | 56 | Non-Organic Insomnia |
| type8 | (Multiple Numbers) | Others |

Table 6.4 The definition of 8 types diagnosis

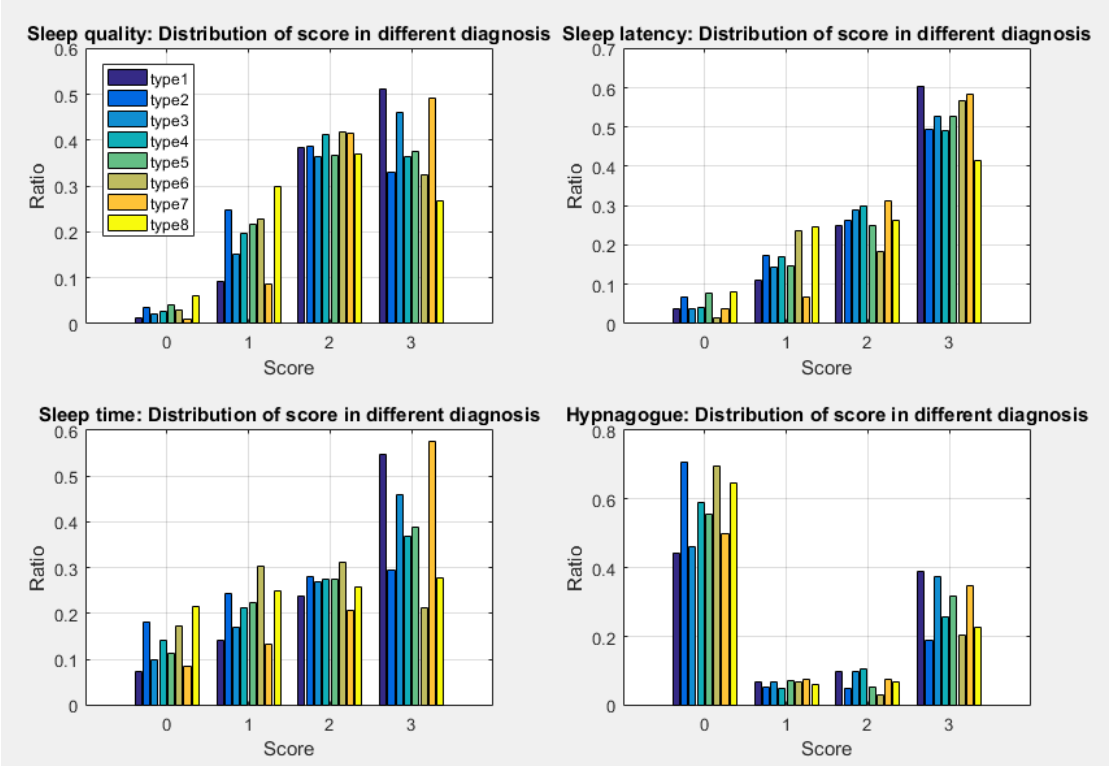


Figure 6.3 The distribution of scores in various sleep condition from problem 2

From the figure above, we can see that no two rectangles with the same type appear

to be with the same height, or specifically speaking, in a certain figure there are always two rectangles with the same type with a height difference of more than 10%. We can draw a conclusion that each sleep condition indicator does affect at least one type of diagnosis of 8 defined types. Therefore, no indicator should be removed from the input of the RDNN.

### 6.3 Training and Parameters Setting

The train and test dataset for RDNN are all from Annex II. We use Momentum optimizer [2] and take the advantage of 100,000 iterated times to minimize the objective function (1). The dataset is randomly divided into 2 kinds of data: 80% train data and 20% test data.

### 6.4 Result

The model result, which is the answer to the problem 3 is shown below in table 6.5. For simplicity, several columns are omitted. A full result can be found in attachments.

| Number | Age | Sex | Source | Predicated Diagnosis | Probability |
|---|---|---|---|---|---|
| 1 | 28 | male | Outpatient | Depression | 61.80% |
| 2 | 37 | male | Outpatient | Depression | 72.30% |
| 3 | 45 | male | Outpatient | Sleep disorder | 92.50% |
| 4 | 32 | female | Outpatient | Sleep disorder | 83.10% |
| 5 | 64 | male | Outpatient | Depression | 51.20% |
| 6 | 29 | female | Outpatient | Sleep disorder | 91.70% |
| 7 | 42 | female | Outpatient | Depression | 61.00% |
| 8 | 36 | female | Outpatient | Sleep disorder | 54.50% |
| 9 | 71 | female | Outpatient | Mixed anxiety and Depression | 94.30% |
| 10 | 26 | female | Outpatient | Sleep disorder | 74.30% |

Table 6.5 Modeling result: answer to the problem 3

### 6.5 Model Validation

To evaluate the accuracy of RDNN, 20% of the data are tested on two other classification method SVM [3] and single classifier, in comparison with RDNN. The

learning rate of Momentum optimizer is set to $10^{-3}$, the momentum 0.01, and the mini-batch size is set to 20.

| Method | SVM | Single classifier | RDNN | |
|---|---|---|---|---|
| | | | RDNN1 | RDNN2 |
| Accuracy | 45% | 55% | 89% | 86% |

Table 6.6 The accuracy result amid SVM, Single classifier and RDNN

## 7. Model Establishment and Solution for Problem 4

### 7.1 Analysis of the Current Situation

*National Sleep Foundation* (NSF) have raised people's attention on the way they sleep and rest.

Given the situation that we construct a RDNN model to solve the problem 2, we can now give patient diagnosis and predicate which diseases they possibly have. In order to help people's recovery, it's essential to develop a strategy to give people advice on how they sleep or rest. After researching and data collecting, we believe that besides the indicators mentioned in original data, bedtime routine (such as staying up late, working overtime constantly, being night workers etc.) may also influence our sleep quality and health.

### 7.2 The Workflow of Our Sleep Program

Before applying the RDNN model, we design a questionnaire to collect information about people's sleep conditions as indicators:

| Corresponding score / Indicator | 0 | 1 | 2 | 3 | Remarks |
|---|---|---|---|---|---|
| Sleep quality | 0 | 1 | 2 | 3 | |
| Sleep latency (mins) | <15 | 15-30 | 30-60 | >60 | |
| Sleep time (hours) | >8 | 6-8 | 4-6 | <4 | |

| | | | | | |
|---|---|---|---|---|---|
| Sleep efficiency | $\text{Eff} = \frac{\frac{1}{score(d)+1}}{\frac{1}{score(t)+1}} \times 100\% = \frac{score(t)+1}{score(d)+1} \times 100\%$<br>$\text{score(e)} = f(e) = \begin{cases} 3, & Eff \leq 0.25 \\ 2, & 0.25 < Eff \leq 0.5 \\ 1, & 0.5 < Eff \leq 0.75 \\ 0, & 0.75 < Eff \leq 1 \end{cases}$ | | | | |
| Sleep disorder | Never | Sometimes | Usually | Always | |
| Hypnagogue | Never | Sometimes | Usually | Always | |
| Dysfunction | Active | Need a nap | Sleepy | Bad | |

Table 7.1 The questionnaire to collect people's sleep conditions

(d: Dysfunction, t: Sleep time)

The score of sleep latency is judged by the average time it takes for people to fall asleep, and the one of sleep time by the average time that people are truly asleep. In consideration of the fact that NSF defines insomnia as the status of sleeplessness for more than 30 minutes, the time interval we set on the questionnaire is reasonable and proper.

It's more appropriate for several indicators to be shown in friendly words rather than fixed numeric level. For example, in *sleep disorder* and *hypnagogue*, four ranks are corresponding with four words (Never, Sometimes, Usually, Always). Besides, the score of sleep efficiency cannot be obtained directly because people themselves have no idea about the way to measure their sleep efficiency, for our program not providing exact criteria. Therefore, calculating it by sleep time and dysfunction provided by people will be an effective choice. Through our life experience that the more people sleep, the more energetic they are, it's certain that if someone has a relatively long sleep time but usually behave inactively, his sleep efficiency will be relatively low. The sleep efficiency formula follows the above rule.

After collecting information about people's sleep condition, we can apply them on the RDNN model and give a general prediction. The structure of our program is shown below in figure 7.1, with the upper layer as a user interface and the lower layer as a dominant inner model prediction layer. This program gives people advice on improving bedtime routine and helps them recover from diseases.

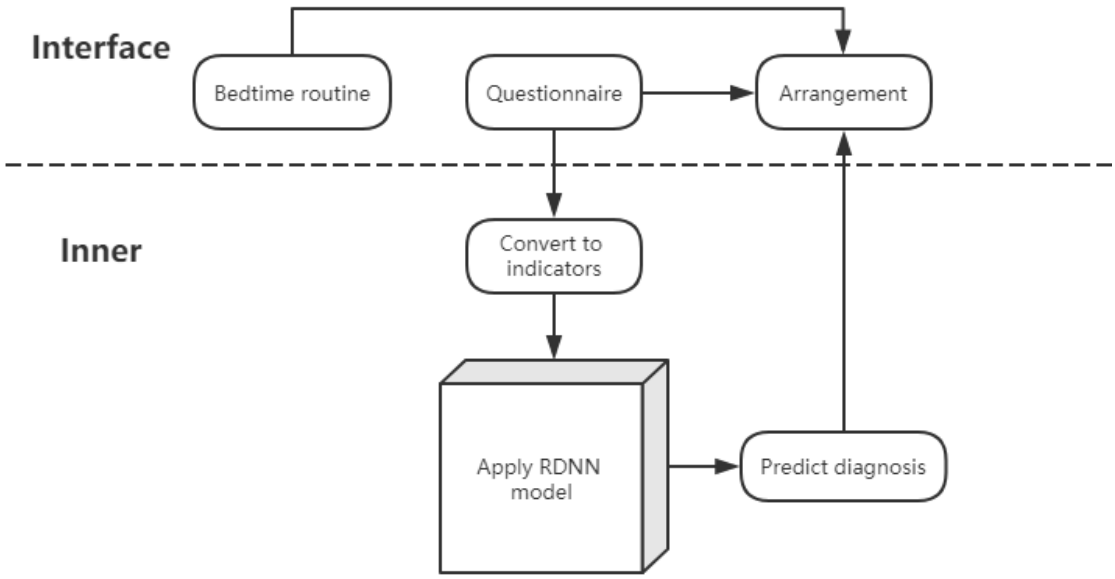


Figure 7.1 The Structure of Our Sleep Program

### 7.3 Suggestions for Improving Bedtime Routines

From Figure 6.3 in modelling for problem 2 we can draw a conclusion that the lower the sleep latency, the longer the sleep time, the better the sleep quality and the lower the risk of having diseases. In terms of hypnagogue, the people who never take medicine or take excessive medicine have a high risk of having diseases, while the risk of it for people with moderate dose of medicine is relatively low.

To scientifically arrange our bedtime routines, several effective suggestions are given. For people who want less sleeping time, drink milk and use less electronic devices. For those who don't, sleep earlier at night. For people who have a continuous low sleep quality, taking medicine with moderate dose to help fall asleep is suggested.

### 7.4 The Evaluation of Effectiveness

Accuracy and reasonableness are considered to evaluate the program. Accuracy derives from the prediction and judgement of RDNN model. In the model of problem 2, we classify the diagnoses by their occurrence rate. 7 diagnoses with the highest occurrence rates are classified to 7 class correspondingly as the input to the RDNN model due to the fact that they occupy more than 85% of all the diagnoses, and the rest are classified to the $8^{th}$ class. The RDNN model has an accuracy rate at approximately

86% in the test of historic data. According to the analysis, there is considerable data stew in the given data, indicating that most patients are suffering from these 7 types of diseases. From what we have mentioned above we can draw a conclusion that RDNN model has a relatively high accuracy for the common diseases. As for reasonableness, the advice for people's rest are restricted by their bedtime routine, ensuring that a reasonable arrangement is given. Different from the program which only gives a therapeutic schedule to a certain disease, our program excludes suggestions that are not appropriate for certain patients and gives advice that suits them best.

## 8. Advantages and Disadvantages

### 8.1 Advantages

The model uses the knowledge of deep learning and it performs well in data sets. Adopting a brand-new neural networks structure, though only a limited amount of data is given and indicators are too few, the model still gives a result with high accuracy.

### 8.2 Disadvantages

The model can only predict single disease. For prediction of mixed diseases, our model can't help. Also, the amount of disease is limited. To tackle this problem, we classify the diseases and divide them into several groups for better prediction.

## References

[1] Krizhevsky, Alex, Sutskever I and Hinton G E, ImageNet classification with deep convolutional neural networks, Annual Conf. on Neural Information Processing Systems, 25, 1097-1105, 2012.

[2] Sutskever I, Martens J, Dahl G, et al, On the importance of initialization and momentum in deep learning, International Conference on International Conference on Machine Learning, 28, 1139-1147, 2013.

[3] Luts J, Ojeda F, Van d P R, et al, A tutorial on support vector machine-based methods for classification problems in chemometrics, Analytica Chimica Acta, 665(2), 129, 2010.

[4] Frank R. Giordano, William P. Fox, Steven B. Horton, A First Course in Mathematical Modeling, China Machine Press, 2014.

# Appendix

**Software**: We mainly use matlab and python.

Matlab code：

1. **Draw line chart (problem1.m, cal.m)**

```
clear;
[num,txt,raw] = xlsread('Annex I');
group0 = zeros(1,8);
group1 = group0;
group2 = group0;
group3 = group0;

[row,col] = size(num);
for i = 1 : row
    slpqlty = num(i,4);
    if slpqlty == 0
        group0 = [group0; num(i,:)];
    elseif slpqlty == 1
        group1 = [group1; num(i,:)];
    elseif slpqlty == 2
        group2 = [group2; num(i,:)];
    elseif slpqlty == 3
        group3 = [group3; num(i,:)];
    end
end

% For group0 - group3
i = 1;
titles = {'Reliability: Ratio of each score','Psychoticism: Ratio of each
score','Nervousness: Ratio of each score','Character: Ratio of each score'};
for j = 5:8
    X = group0(:,i+4);
    cnt_X = cal(X);
    figure();
    plot(1:100,cnt_X,'r');
    str{1}=['group0'];
    hold on;
    grid on;

    X = group1(:,i+4);
    cnt_X = cal(X);
```

```
        plot(1:100,cnt_X,'g');
        str{2}=['group1'];

        X = group2(:,i+4);
        cnt_X = cal(X);
        plot(1:100,cnt_X,'b');
        str{3}=['group2'];

        X = group3(:,i+4);
        cnt_X = cal(X);
        plot(1:100,cnt_X,'k');
        str{4}=['group3'];

        title(titles(i));
        xlabel('Indicator');
        ylabel('Ratio')
        i = i + 1;

        legend(str);
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% cal.m
    function [ cnt_X ] = cal( X )
    %UNTITLED2 Summary of this function goes here
    %     Detailed explanation goes here

    [wid,len] = size(X');
    cnt_X = zeros(1,100);
    for i = 2:len
        j = round(X(i)) + 1;
        cnt_X(j) = cnt_X(j) + 1;
    end

    for i = 1:100
        cnt_X(i) = cnt_X(i) / len;
    end
    end
```

2. **Draw line chart and bar chart (after classifying) (problem1_bar_chart.m, cal_bar_chart.m)**

```
clear;
```

```
[num,txt,raw] = xlsread('Annex I');
group0 = zeros(1,8);
group1 = group0;
group2 = group0;
group3 = group0;

[row,col] = size(num);
for i = 1 : row
    slpqlty = num(i,4);
    if slpqlty == 0
        group0 = [group0; num(i,:)];
    elseif slpqlty == 1
        group1 = [group1; num(i,:)];
    elseif slpqlty == 2
        group2 = [group2; num(i,:)];
    elseif slpqlty == 3
        group3 = [group3; num(i,:)];
    end
end

% Draw
i = 1;
titles = {'Reliability: Ratio of each section','Psychoticism: Ratio of each
section','Nervousness: Ratio of each section','Character: Ratio of each
section'};
vv = zeros(1,10);
for i = 1:4
    ss = zeros(1, 10)'; %sum
    X = group0(:,i+4);
    cnt_X = cal_bar_chart(X);
    ss = [ss cnt_X(1,:)'];
    figure();
    plot(1:10, cnt_X, 'r');
    str{i}=['group0'];
    hold on;
    grid on;

    X = group1(:,i+4);
    cnt_X = cal_bar_chart(X);
    ss = [ss cnt_X(1,:)'];
    plot(1:10,cnt_X,'g');
```

```
        str{2}=['group1'];

        X = group2(:,i+4);
        cnt_X = cal_bar_chart(X);
        ss = [ss cnt_X(1,:)'];
        plot(1:10,cnt_X,'b');
        str{3}=['group2'];

        X = group3(:,i+4);
        cnt_X = cal_bar_chart(X);
        ss = [ss cnt_X(1,:)'];
        plot(1:10,cnt_X,'k');
        str{4}=['group3'];

        title(titles(i));
        xlabel('Indicator');
        ylabel('Ratio');
        legend(str);

        ss(:,1)=[];
        v = [0];
        for p = 1:10
            temp = var(ss(p,:).*100, 1);
            v = [v temp];
        end
        v(:,1)=[];
        vv = [vv; v];
end
vv(1,:) = [];
str_bar = {'0-10','10-20','20-30','30-40','40-50','50-60','60-70','70-80','80-
90','90-100'};
figure();
bar(vv);
title('Comparison of variance in different sections (Enlarged)');
xlabel('Score of each indicator');
ylabel('Variance');
legend(str_bar);
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cal_bar_chart.m
```

```
function [ cnt_X ] = cal_bar_chart( X )
%UNTITLED2 Summary of this function goes here
%     Detailed explanation goes here
%     Divided 100 into 10 parts

[wid,len] = size(X');
cnt_X = zeros(1,10);
for i = 2:len
      j = floor(X(i)/10)+1;
      cnt_X(j) = cnt_X(j) + 1;
end
for i = 1:10
      cnt_X(i) = cnt_X(i) / len;
end
end
```

**3. Draw heatmap (hmap.m)**

```
clear;
[num,txt,raw] = xlsread('Annex I');
group0 = [];
group1 = group0;
group2 = group0;
group3 = group0;

[row,col] = size(num);
for i = 1 : row
      slpqlty = num(i,4);
      if slpqlty == 0
            group0 = [group0; num(i,:)];
      elseif slpqlty == 1
            group1 = [group1; num(i,:)];
      elseif slpqlty == 2
            group2 = [group2; num(i,:)];
      elseif slpqlty == 3
            group3 = [group3; num(i,:)];
      end
end

tlb = [];
for i = 1:4
      X = group0(:,i+4);
```

```
        cnt_X = cal_bar_chart(X);
        tlb = [tlb;cnt_X];

        X = group1(:,i+4);
        cnt_X = cal_bar_chart(X);
        tlb = [tlb;cnt_X];

        X = group2(:,i+4);
        cnt_X = cal_bar_chart(X);
        tlb = [tlb;cnt_X];

        X = group3(:,i+4);
        cnt_X = cal_bar_chart(X);
        tlb = [tlb;cnt_X];
end
tR = tlb(1:4,:);
tP = tlb(5:8,:);
tN = tlb(9:12,:);
tC = tlb(13:16,:);
xvar = [1,2,3,4,5,6,7,8,9,10];
yvar = [0,1,2,3];

% redgreencmap
% redbluecmap*255
% myColorcMap = redbluecmap + 0.02;

hm = HeatMap(tR,'ColumnLabels',xvar,'RowLabels',yvar,
'Annotate',true,'Colormap', redbluecmap, 'ColumnLabelsRotate', 0,
'LabelsWithMarkers', true);
addXLabel(hm, 'Feature distribution');
addYLabel(hm, 'Sleep quality');
addTitle(hm, 'Reliability: Distribution of indicators');

hm = HeatMap(tP,'ColumnLabels',xvar,'RowLabels',yvar,
'Annotate',true,'Colormap', redbluecmap,    'ColumnLabelsRotate', 0);
addXLabel(hm, 'Feature distribution');
addYLabel(hm, 'Sleep quality');
addTitle(hm, 'Psychoticism: Distribution of indicators');

hm = HeatMap(tN,'ColumnLabels',xvar,'RowLabels',yvar,
'Annotate',true,'Colormap', redbluecmap,    'ColumnLabelsRotate', 0);
```

```
addXLabel(hm, 'Feature distribution');
addYLabel(hm, 'Sleep quality');
addTitle(hm, 'Nervousness: Distribution of indicators');

hm = HeatMap(tC,'ColumnLabels',xvar,'RowLabels',yvar,
'Annotate',true,'Colormap', redbluecmap,    'ColumnLabelsRotate', 0);
addXLabel(hm, 'Feature distribution');
addYLabel(hm, 'Sleep quality');
addTitle(hm, 'Charater: Distribution of indicators');
```

**4. Draw bar chart of age and sex (AgeSex.m)**

```
clear;
[num, txt, raw] = xlsread('Annex I.xlsx');
[row, col] = size(raw);

% Convert male and female to 0 and 1
for i = 2:row
    if strcmp(raw(i, 3), 'male') == 1
        num(i-1, 2) = 0;
    elseif strcmp(raw(i, 3), 'female') == 1
        num(i-1, 2) = 1;
    end
end
row = row - 1;

% Get data
data = [num(:,1) num(:,2) num(:,4)]; % age - sex - quality
sex = zeros(2,4);
for s = 1:row
    qua = data(s, 3);
    gender = data(s, 2);
    sex(gender+1, qua+1) = sex(gender+1, qua+1) + 1;
end

% Draw gender
sex(1,:) = sex(1,:) ./ sum(sex(1,:));
sex(2,:) = sex(2,:) ./ sum(sex(2,:));
str_sex = {'male','female'};
bar(sex');
grid on;
title('Distribution of sleep quality in different gender');
```

```
xlabel('Sleep quality');
ylabel('Ratio');
legend(str_sex,'Location','NorthWest');
set(gca,'xticklabel',0:1:4);

% Draw variance
varianceOfSex = [];
for k = 1:4
    temp = var(sex(:,k).*100, 1);
    varianceOfSex = [varianceOfSex temp];
end
figure();
color=[10 10 10];
bar(varianceOfSex);
grid on;
% axis([0 1 0 5]);
set(gca,'xticklabel',0:1:4);
set(gca,'yticklabel',0:30);
title('Variance of distribution (Enlarged)');
xlabel('Sleep quality')
ylabel('Variance');


% Age
young = [];
middle = [];
elder = [];
old = [];
for i = 1:row
    age = data(i, 1);
    if age <= 44
        young = [young; data(i,:)];
    elseif age <= 59
        middle = [middle; data(i,:)];
    elseif age <= 74
        elder = [elder; data(i,:)];
    elseif age <= 89
        old = [old; data(i,:)];
    end
end
```

```
% Calculate the ratio of each age group
cnt = zeros(4, 4);
% young
[len, wid] = size(young);
for j = 1:len
    qua = young(j, 3);
    cnt(1, qua+1) = cnt(1, qua+1) + 1;
end
cnt(1,:) = cnt(1,:) ./ len;

% middle
[len, wid] = size(middle);
for j = 1:len
    qua = middle(j, 3);
    cnt(2, qua+1) = cnt(2, qua+1) + 1;
end
cnt(2,:) = cnt(2,:) ./ len;

% elder
[len, wid] = size(elder);
for j = 1:len
    qua = elder(j, 3);
    cnt(3, qua+1) = cnt(3, qua+1) + 1;
end
cnt(3,:) = cnt(3,:) ./ len;

% old
[len, wid] = size(old);
for j = 1:len
    qua = old(j, 3);
    cnt(4, qua+1) = cnt(4, qua+1) + 1;
end
cnt(4,:) = cnt(4,:) ./ len;


% Draw age
str_age = {'<44','45-59','60-74','>75'};
figure();
bar(cnt');
set(gca,'xticklabel',0:1:10);
grid on;
```

```
legend(str_age,'Location','NorthWest');
title('Distribution of sleep quality in each age group');
xlabel('Sleep quality');
ylabel('Ratio');

% Draw variance of age
varianceOfAge = [];
for k = 1:4
    temp = var(cnt(:,k)' .* 100, 1);
    varianceOfAge = [varianceOfAge temp];
end
% str_varianceOfAge = {};
figure();
bar(varianceOfAge);
set(gca,'xticklabel',0:1:4);
grid on;
% legend(str_varianceOfAge,'Location','NorthWest');
title('Variance of distribution (Enlarged)');
xlabel('Sleep quality')
ylabel('Variance');
```

**5. Visualize the data in Annex II(problem2_preprocessing.m)**

```
clear;
sort_data = xlsread('2-1.xlsx',2,'B3:C131');
[row, col] = size(sort_data);
bar(sort_data(:,2));
title('Number of each diagnosis');
xlabel('Diagnosis');
ylabel('Number');
set(gca,'xticklabel');
for i = 1:row
    if isnan(sort_data(i,1))
        sort_data(i,:) = [];
    end
end

data = csvread('D:/data.csv');
[row,col] = size(data);
for i = 1:row
    d = data(i,1);
    if d == 57
        data(i,1) = 1;
```

```
        elseif d == 33
            data(i,1) = 2;
        elseif d == 6
            data(i,1) = 3;
        elseif d == 5
            data(i,1) = 4;
        elseif d == 74
            data(i,1) = 5;
        elseif d == 15
            data(i,1) = 6;
        elseif d == 56
            data(i,1) = 7;
        else
            data(i,1) = 8;
        end
    end

    input = data(:,2:8);
    output = data(:,1);
    cnt_qua = zeros(8,4);
    cnt_lat = cnt_qua;
    cnt_tme = cnt_qua;
    cnt_eff = cnt_qua;
    cnt_dis = cnt_qua;
    cnt_hyn = cnt_qua;
    cnt_fnc = cnt_qua;
    num = zeros(1,8);

    % Processing
    for i = 1:row
        type = output(i,:);
        num(1,type) = num(1,type) + 1;
        cnt_qua(type, input(i, 1)+1) = cnt_qua(type, input(i, 1)+1) + 1;
        cnt_lat(type, input(i, 2)+1) = cnt_lat(type, input(i, 2)+1) + 1;
        cnt_tme(type, input(i, 3)+1) = cnt_tme(type, input(i, 3)+1) + 1;
        cnt_eff(type, input(i, 4)+1) = cnt_eff(type, input(i, 4)+1) + 1;
        cnt_dis(type, input(i, 5)+1) = cnt_dis(type, input(i, 5)+1) + 1;
        cnt_hyn(type, input(i, 6)+1) = cnt_hyn(type, input(i, 6)+1) + 1;
        cnt_fnc(type, input(i, 7)+1) = cnt_fnc(type, input(i, 7)+1) + 1;
    end
```

```
% Calculate the rate
for i = 1:8
    cnt_qua(i,:) = cnt_qua(i,:) ./ num(1,i);
    cnt_lat(i,:) = cnt_lat(i,:) ./ num(1,i);
    cnt_tme(i,:) = cnt_tme(i,:) ./ num(1,i);
    cnt_eff(i,:) = cnt_eff(i,:) ./ num(1,i);
    cnt_dis(i,:) = cnt_dis(i,:) ./ num(1,i);
    cnt_hyn(i,:) = cnt_hyn(i,:) ./ num(1,i);
    cnt_fnc(i,:) = cnt_fnc(i,:) ./ num(1,i);
end

str = {'type1','type2','type3','type4','type5','type6','type7','type8'};
% Draw
figure();
bar(cnt_qua');
set(gca,'xticklabel',0:1:4);
grid on;
title('Sleep quality: Distribution of score in different diagnosis')
xlabel('Score');
ylabel('Ratio');
legend(str,'Location','NorthWest');

figure();
bar(cnt_lat');
set(gca,'xticklabel',0:1:4);
grid on;
title('Sleep latency: Distribution of score in different diagnosis')
xlabel('Score');
ylabel('Ratio');
legend(str,'Location','NorthWest');

figure();
bar(cnt_tme');
set(gca,'xticklabel',0:1:4);
grid on;
title('Sleep time: Distribution of score in different diagnosis')
xlabel('Score');
ylabel('Ratio');
legend(str,'Location','NorthWest');

figure();
```

```
bar(cnt_eff');
set(gca,'xticklabel',0:1:4);
grid on;
title('Sleep efficiency: Distribution of score in different diagnosis')
xlabel('Score');
ylabel('Ratio');
legend(str,'Location','NorthWest');

figure();
bar(cnt_dis');
set(gca,'xticklabel',0:1:4);
grid on;
title('Sleep disorder: Distribution of score in different diagnosis')
xlabel('Score');
ylabel('Ratio');
legend(str);

figure();
bar(cnt_hyn');
set(gca,'xticklabel',0:1:4);
grid on;
title('Hypnagogue: Distribution of score in different diagnosis')
xlabel('Score');
ylabel('Ratio');
legend(str);

figure();
bar(cnt_fnc');
set(gca,'xticklabel',0:1:4);
grid on;
title('Daytime dyfunction: Distribution of score in different diagnosis')
xlabel('Score');
ylabel('Ratio');
legend(str);
```

**Python code:**

1. **network_1_classifier.py**

```
# -*- coding: utf-8 -*-
from __future__ import division
import tensorflow as tf
```

```
import os
import numpy as np
import xlrd
from read_csv_1_classifier import *
import csv

def get_weights(shape, name=None):
    weight_initializer = tf.contrib.layers.xavier_initializer()
    W = tf.get_variable(name, shape, tf.float32, weight_initializer)
    return W

def bias_variable(shape):
    initial = tf.constant(0.1, shape=shape)
    return tf.Variable(initial)


xs = tf.placeholder(tf.float32, shape = (None, 7)) # 20 维  6 个数据
ys = tf.placeholder(tf.float32, shape = (None, 8))
#
keep_prob = tf.placeholder(tf.float32)
xs = tf.reshape(xs, [-1, 7])
ys = tf.reshape(ys, [-1, 8])

w_fc1 = get_weights([7, 128], name='w_fc1')
b_fc1 = bias_variable([128])
h_fc1 = tf.nn.softmax(tf.matmul(xs, w_fc1) + b_fc1)
h_fc1 = tf.nn.dropout(h_fc1, keep_prob)

w_fc2 = get_weights([128, 64], name='w_fc2')
b_fc2 = bias_variable([64])
h_fc2 = tf.nn.softmax(tf.matmul(h_fc1, w_fc2) + b_fc2)
h_fc2 = tf.nn.dropout(h_fc2, keep_prob)

w_fc3 = get_weights([64, 32], name='w_fc3')
b_fc3 = bias_variable([32])
h_fc3 = tf.nn.softmax(tf.matmul(h_fc2, w_fc3) + b_fc3)
```

```
h_fc3 = tf.nn.dropout(h_fc3, keep_prob)

w_fc7 = get_weights([32, 8], name='w_fc7')
b_fc7 = bias_variable([8])
pred_o = tf.nn.softmax(tf.matmul(h_fc3, w_fc7) + b_fc7)

cross_entropy = -tf.reduce_sum(ys * tf.log(pred_o))

train_step = tf.train.MomentumOptimizer(0.001,
0.01).minimize(cross_entropy)
pred_1 = tf.argmax(pred_o, 1)
pred_2 = tf.argmax(ys, 1)

correct_pred = tf.equal(tf.argmax(pred_o, 1), tf.argmax(ys, 1))

accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

init = tf.global_variables_initializer()

xss, yss, validation_x, validation_y = getdata()

i = 0
j = 0
sum = 0
with tf.Session() as sess:
    sess.run(init)
    for epoch in xrange(1000000):
        #x_s, y_s = sess.run([xss, yss])

        loss, _, p4= sess.run([cross_entropy, train_step, pred_o],
feed_dict={xs: xss[i], ys: yss[i], keep_prob:1})
        #print p.shape
        i += 1
        if (i == 279):
            i = 0
```

```
            #print loss
            if (epoch % 100 == 0):
                p, p1, p2 = sess.run([pred_o, pred_1, pred_2], feed_dict={xs:
validation_x, ys: validation_y, keep_prob: 1.0})
                train_accuracy = accuracy.eval(feed_dict={xs: validation_x, ys:
validation_y, keep_prob:1.0})
                #print p
                #for i in range(20):
                #    print sum
                #    sum = 0
                #    for j in range(8):
                #        sum = sum + p[i][j]
                print p1
                print p2
                #print validation_y
                j+=1
                if (j == 99):
                    j = 0
                #print validation_y.shape
                print 'step %d, training accuracy %g' % (epoch, train_accuracy)
sess.close()
```

**2. RDNN1.py**

```
# -*- coding: utf-8 -*-
from __future__ import division
import tensorflow as tf
import os
import numpy as np
import xlrd
import csv
from read_RDNN import *
def get_weights(shape, name):
    weight_initializer = tf.contrib.layers.xavier_initializer()
    W = tf.get_variable(name, shape, tf.float32, weight_initializer)
    return W
```

```
def bias_variable(shape, name):
    initial = tf.constant(0., shape=shape)
    return tf.Variable(initial, name)


xs = tf.placeholder(tf.float32, shape = (None, 7), name='x_input') # 20 维 6
个数据
ys = tf.placeholder(tf.float32, shape = (None, 4), name='y_input')
#
xs = tf.reshape(xs, [-1, 7])
ys = tf.reshape(ys, [-1, 4])

w_fc1 = get_weights([7, 128], name='w_fc1')
b_fc1 = bias_variable([128], name='b_fc1')
h_fc1 = tf.nn.relu(tf.matmul(xs, w_fc1) + b_fc1)
#h_fc1 = tf.nn.dropout(h_fc1, keep_prob)

w_fc2 = get_weights([128, 64], name='w_fc2')
b_fc2 = bias_variable([64], name='b_fc2')
h_fc2 = tf.nn.relu(tf.matmul(h_fc1, w_fc2) + b_fc2)
#h_fc2 = tf.nn.dropout(h_fc2, keep_prob)

w_fc3 = get_weights([64, 32], name='w_fc3')
b_fc3 = bias_variable([32], name='b_fc3')
h_fc3 = tf.nn.relu(tf.matmul(h_fc2, w_fc3) + b_fc3)
#h_fc3 = tf.nn.dropout(h_fc3, keep_prob)

w_fc4 = get_weights([32, 4], name='w_fc4')
b_fc4 = bias_variable([4], name='b_fc4')
pred_o = tf.nn.softmax(tf.matmul(h_fc3, w_fc4) + b_fc4)

cross_entropy = -tf.reduce_sum(ys * tf.log(pred_o))

#train_step = tf.train.AdamOptimizer(0.001).minimize(cross_entropy)
```

```
train_step = tf.train.MomentumOptimizer(0.001,
0.01).minimize(cross_entropy)
pred_1 = tf.argmax(pred_o, 1)
pred_2 = tf.argmax(ys, 1)

correct_pred = tf.equal(tf.argmax(pred_o, 1), tf.argmax(ys, 1))

accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

init = tf.global_variables_initializer()

xss, yss, validation_x, validation_y, data3 = getlevel()

i = 0
j = 0
sum = 0

saver = tf.train.Saver(max_to_keep=20)

with tf.Session() as sess:
    sess.run(init)
    for epoch in xrange(200000):
        #x_s, y_s = sess.run([xss, yss])
        loss, _, p4= sess.run([cross_entropy, train_step, pred_o],
feed_dict={xs: xss[i], ys: yss[i]})
        #print p.shape
        #print loss
        i += 1
        if (i == 285):
            i = 0
        #print loss
        if (epoch % 100 == 0):
            p, p1, p2 = sess.run([pred_o, pred_1, pred_2], feed_dict={xs:
validation_x, ys: validation_y})
```

```
                train_accuracy = accuracy.eval(feed_dict={xs: validation_x, ys:
validation_y})
                #print p
                #for i in range(20):
                #       print sum
                #       sum = 0
                #       for j in range(8):
                #               sum = sum + p[i][j]
                #print p1
                #print p2
                #print validation_y
                j+=1
                if (j == 99):
                        j = 0
                #print validation_y.shape
                print 'step %d, training accuracy %g' % (epoch, train_accuracy)
                if (train_accuracy >= 0.8):
                        p_final, b= sess.run([pred_o, pred_1], feed_dict={xs:
data3})
                        print p_final
                        print b

                        #saver.save(sess,
save_path='./network1_sessdata/save_net.ckpt', global_step=epoch)
                        #print 'Save check point %d' % epoch
sess.close()
```

**3. RDNN2.py**

```
# -*- coding: utf-8 -*-
from __future__ import division
import tensorflow as tf
import os
import numpy as np
import xlrd
import csv
```

```
from read_RDNN import *

def get_weights(shape, name):
    weight_initializer = tf.contrib.layers.xavier_initializer()
    W = tf.get_variable(name, shape, tf.float32, weight_initializer)
    return W


def bias_variable(shape, name):
    initial = tf.constant(0., shape=shape)
    return tf.Variable(initial, name)


xs = tf.placeholder(tf.float32, shape=(None, 7), name='x_input')  # 20 维 6
个数据
ys = tf.placeholder(tf.float32, shape=(None, 5), name='y_input')
#
xs = tf.reshape(xs, [-1, 7])
ys = tf.reshape(ys, [-1, 5])

w_fc1 = get_weights([7, 128], name='w_fc1')
b_fc1 = bias_variable([128], name='b_fc1')
h_fc1 = tf.nn.relu(tf.matmul(xs, w_fc1) + b_fc1)
# h_fc1 = tf.nn.dropout(h_fc1, keep_prob)

w_fc2 = get_weights([128, 64], name='w_fc2')
b_fc2 = bias_variable([64], name='b_fc2')
h_fc2 = tf.nn.relu(tf.matmul(h_fc1, w_fc2) + b_fc2)
# h_fc2 = tf.nn.dropout(h_fc2, keep_prob)

w_fc3 = get_weights([64, 32], name='w_fc3')
b_fc3 = bias_variable([32], name='b_fc3')
h_fc3 = tf.nn.relu(tf.matmul(h_fc2, w_fc3) + b_fc3)
# h_fc3 = tf.nn.dropout(h_fc3, keep_prob)
```

```
w_fc4 = get_weights([32, 5], name='w_fc4')
b_fc4 = bias_variable([5], name='b_fc4')
pred_o = tf.nn.softmax(tf.matmul(h_fc3, w_fc4) + b_fc4)

cross_entropy = -tf.reduce_sum(ys * tf.log(pred_o))

# train_step = tf.train.AdamOptimizer(0.001).minimize(cross_entropy)
train_step = tf.train.MomentumOptimizer(0.001,
0.01).minimize(cross_entropy)
pred_1 = tf.argmax(pred_o, 1)
pred_2 = tf.argmax(ys, 1)

correct_pred = tf.equal(tf.argmax(pred_o, 1), tf.argmax(ys, 1))

accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

init = tf.global_variables_initializer()

xss, yss, validation_x, validation_y, data3 = get_5_scale()
v_x = np.zeros([1, 7])
i = 0
j = 0
sum = 0

saver = tf.train.Saver(max_to_keep=20)

with tf.Session() as sess:
    sess.run(init)
    for epoch in xrange(100000):
        # x_s, y_s = sess.run([xss, yss])
        loss, _, p4 = sess.run([cross_entropy, train_step, pred_o],
feed_dict={xs: xss[i], ys: yss[i]})
        # print p.shape
        i += 1
```

```
            if (i == 89):
                i = 0
            # print loss
            if (epoch % 100 == 0):
                p, p1, p2 = sess.run([pred_o, pred_1, pred_2], feed_dict={xs:
validation_x, ys: validation_y})
                train_accuracy = accuracy.eval(feed_dict={xs: validation_x, ys:
validation_y})
                # print p
                # for i in range(20):
                #       print sum
                #       sum = 0
                #       for j in range(8):
                #             sum = sum + p[i][j]
                #print p1
                #print p2
                # print validation_y
                j += 1
                if (j == 99):
                    j = 0
                # print validation_y.shape
                print 'step %d, training accuracy %g' % (epoch, train_accuracy)
                if (train_accuracy >= 0.8):
                    #path = saver.save(sess, "./network_sessdata/model.md")
                    p0_s, p1_s= sess.run([pred_o, pred_1], feed_dict={xs:
data3})
                    print p1_s

                    #saver.save(sess,
save_path='./network1_sessdata/save_net.ckpt', global_step=epoch)
                    print 'Save check point %d' % epoch

sess.close()
```

**4. read_csv_1_classifier.py**

```
# -*- coding: utf-8 -*-
from __future__ import division
# 5 6 15 33 56 57 74
import csv
import numpy as np
common = ['5', '6', '15', '33', '56', '57', '74']


def standarize3(index):
    count = common.index(index)
    return [0] * count + [1] + [0] * (len(common) - count)

def standarize(index): # max 122
    common = ['5', '6', '15', '33', '56', '57', '74']
    if index in common:
        count = common.index(index)
        return [0]*count + [1] + [0]*(len(common)-count)
    else:
        return [0]*(len(common)) + [1]

def getdata():
    csv_reader = csv.reader(open('data_1_classifier.csv'))
    input_array = []
    output_array = []
    for row in csv_reader:
        input_array.append(map(lambda x: int(x), row[1:8]))#7
        output_array.append(standarize(row[0]))#8 0000100

    input_x = np.zeros([5753, 7])
    for i in range(input_x.shape[0]):
        for x in range(input_x.shape[1]):
            input_x[i][x] = input_array[i][x]

    I = np.zeros([280, 20, 7])
    count1 = -1
```

```
    for i in range(I.shape[0]):
        for x in range(I.shape[1]):
            count1+=1
            for j in range(I.shape[2]):
                I[i][x][j] = input_x[count1][j]

    input_y = np.zeros([5753, 8])
    for i in range(input_y.shape[0]):
        for x in range(input_y.shape[1]):
            input_y[i][x] = output_array[i][x]

    O = np.zeros([280, 20, 8])
    count2 = -1
    for i in range(O.shape[0]):
        for x in range(O.shape[1]):
            count2 += 1
            for j in range(O.shape[2]):
                O[i][x][j] = input_y[count2][j]


    validation_I = np.zeros([100, 7])
    for i in range(100):
        for j in range(7):
            validation_I[i][j] = input_x[2000+i][j]
    validation_O = np.zeros([100, 8])
    for i in range(100):
        for j in range(8):
            validation_O[i][j] = input_y[2000+i][j]

    return I, O, validation_I, validation_O



def getdata2():
    csv_reader = csv.reader(open('data_1_classifier.csv'))
```

```
input_array = []
output_array = []
for row in csv_reader:
    input_array.append(map(lambda x: int(x), row[1:8]))#7
    output_array.append(standarize(row[0]))#8 0000100



input_x = np.zeros([5600, 7])
for i in range(input_x.shape[0]):
    for x in range(input_x.shape[1]):
        input_x[i][x] = input_array[i][x]

I = np.zeros([5600, 1, 7])

for i in range(I.shape[0]):#5753
    for j in range(I.shape[2]):
        I[i][0][j] = input_x[i][j]

input_y = np.zeros([5600, 8])
for i in range(input_y.shape[0]):
    for x in range(input_y.shape[1]):
        input_y[i][x] = output_array[i][x]

O = np.zeros([5600, 1, 8])

for i in range(O.shape[0]):
    for j in range(O.shape[2]):
        O[i][0][j] = input_y[i][j]



validation_I = np.zeros([100, 7])
for i in range(100):
    for j in range(7):
        validation_I[i][j] = input_array[4000+i][j]
```

```
    validation_O = np.zeros([100, 8])
    for i in range(100):
        for j in range(8):
            validation_O[i][j] = output_array[4000+i][j]


    return I, O, validation_I, validation_O

def get_data3():
    csv_reader = csv.reader(open('data_1_classifier.csv'))
    input_array = []
    output_array = []
    for row in csv_reader:
        if row[0] in common:
            input_array.append(map(lambda x: int(x), row[1:8]))
            output_array.append(standarize3(row[0]))

    input_x = np.zeros([5049, 7])
    for i in range(input_x.shape[0]):
        for x in range(input_x.shape[1]):
            input_x[i][x] = input_array[i][x]

    I = np.zeros([250, 20, 7])
    count1 = -1
    for i in range(I.shape[0]):
        for x in range(I.shape[1]):
            count1+=1
            for j in range(I.shape[2]):
                I[i][x][j] = input_x[count1][j]

    input_y = np.zeros([5049, 7])
    for i in range(input_y.shape[0]):
        for x in range(input_y.shape[1]):
            input_y[i][x] = output_array[i][x]

    O = np.zeros([250, 20, 7])
```

```
    count2 = -1
    for i in range(O.shape[0]):
        for x in range(O.shape[1]):
            count2 += 1
            for j in range(O.shape[2]):
                O[i][x][j] = input_y[count2][j]



    validation_I = np.zeros([20, 7])
    for i in range(20):
        for j in range(7):
            validation_I[i][j] = input_x[4000+i][j]
    validation_O = np.zeros([20, 7])
    for i in range(20):
        for j in range(7):
            validation_O[i][j] = input_y[4000+i][j]

    return I, O, validation_I, validation_O
```

**5. read_csv_svm.py**

```
# -*- coding: utf-8 -*-
from __future__ import division
# 5 6 15 33 56 57 74
import csv
import numpy as np

def standarize(index): # max 122
    common = ['5', '6', '15', '33', '56', '57', '74']
    if index in common:
        count = common.index(index)
        return [0]*count + [1] + [0]*(len(common)-count)
    else:
        return [0]*(len(common)) + [1]

def standarize2(index): # max 122
```

```
    common = ['5', '6', '15', '33', '56', '57', '74']
    if index in common:
        return common.index(index) + 1
    else:
        return len(common) + 1

def getdata():
    csv_reader = csv.reader(open('svm_data.csv'))
    input_array = []
    output_array = []
    for row in csv_reader:
        input_array.append(map(lambda x: int(x), row[1:8]))#7
        output_array.append(standarize(row[0]))#8 0000100

    input_x = np.zeros([5753, 7])
    for i in range(input_x.shape[0]):
        for x in range(input_x.shape[1]):
            input_x[i][x] = input_array[i][x]

    I = np.zeros([280, 20, 7])
    count1 = -1
    for i in range(I.shape[0]):
        for x in range(I.shape[1]):
            count1+=1
            for j in range(I.shape[2]):
                I[i][x][j] = input_x[count1][j]

    input_y = np.zeros([5753, 8])
    for i in range(input_y.shape[0]):
        for x in range(input_y.shape[1]):
            input_y[i][x] = output_array[i][x]

    O = np.zeros([280, 20, 8])
    count2 = -1
    for i in range(O.shape[0]):
```

```
            for x in range(O.shape[1]):
                count2 += 1
                for j in range(O.shape[2]):
                    O[i][x][j] = input_y[count2][j]

    validation_I = np.zeros([20, 7])
    for i in range(20):
        for j in range(7):
            validation_I[i][j] = input_x[5600+i][j]
    validation_O = np.zeros([20, 8])
    for i in range(20):
        for j in range(8):
            validation_O[i][j] = input_y[5600+i][j]

    return I, O, validation_I, validation_O

def getdata2():
    csv_reader = csv.reader(open('svm_data.csv'))
    input_array = []
    output_array = []
    for row in csv_reader:
        input_array.append(map(lambda x: int(x), row[1:8]))#7
        output_array.append(standarize2(row[0]))#8 0000100

    input_x = np.zeros([5000, 7])
    for i in range(input_x.shape[0]):
        for x in range(input_x.shape[1]):
            input_x[i][x] = input_array[i+600][x]

    print input_x
    # I = np.zeros([5600, 1, 7])

    # for i in range(I.shape[0]):#5753
    #         for j in range(I.shape[2]):
    #             I[i][0][j] = input_x[i][j]
```

```
    input_y = np.zeros([5000])
    for i in range(input_y.shape[0]):
        input_y[i] = output_array[600+i]

    print input_y
    # O = np.zeros([5600, 1, 8])

    # for i in range(O.shape[0]):
        # for j in range(O.shape[2]):
            # O[i][0][j] = input_y[i][j]

    validation_I = np.zeros([600, 7])
    for i in range(600):
        for j in range(7):
            validation_I[i][j] = input_array[i][j]

    validation_O = np.zeros([600])
    for i in range(600):
        validation_O[i] = output_array[i]

    return input_x, input_y, validation_I, validation_O
```

**6. read_RDNN.py**

```
# -*- coding: utf-8 -*-
from __future__ import division
# 5 6 15 33 56 57 74
import csv
import numpy as np

common_inner = ['6', '5', '74', '15', '56']


def standarize4_inner(index):
    if index in common_inner:
        count = common_inner.index(index)
```

```
            return [0] * count + [1] + [0] * (len(common_inner) - count - 1)
        else:
            raise Exception('invalid disease index')

def standarize4(index):
    common4 = ['57', '33', '6', '5', '74', '15', '56']
    if index in common4:
        count = common4.index(index)
        if count > 2:
            count = 2
        return [0]*count + [1] + [0]*(3-count)
    else:
        return [0]*3 + [1]

def getlevel():
    csv_reader = csv.reader(open('data_RDNN.csv'))
    input_array = []
    output_array = []
    for row in csv_reader:
        input_array.append(map(lambda x: int(x), row[1:8]))    # 7
        output_array.append(standarize4(row[0]))    # 8 0000100

    input_x = np.zeros([5753, 7])
    for i in range(input_x.shape[0]):
        for x in range(input_x.shape[1]):
            input_x[i][x] = input_array[i][x]

    I = np.zeros([286, 20, 7])
    count1 = -1
    for i in range(I.shape[0]):
        for x in range(I.shape[1]):
            count1 += 1
            for j in range(I.shape[2]):
                I[i][x][j] = input_x[count1][j]
```

```
input_y = np.zeros([5753, 4])
for i in range(input_y.shape[0]):
    for x in range(input_y.shape[1]):
        input_y[i][x] = output_array[i][x]

O = np.zeros([286, 20, 4])
count2 = -1
for i in range(O.shape[0]):
    for x in range(O.shape[1]):
        count2 += 1
        for j in range(O.shape[2]):
            O[i][x][j] = input_y[count2][j]

validation_I = np.zeros([100, 7])
for i in range(100):
    for j in range(7):
        validation_I[i][j] = input_x[2000 + i][j]
validation_O = np.zeros([100, 4])
for i in range(100):
    for j in range(4):
        validation_O[i][j] = input_y[2000 + i][j]

data3 = np.zeros([10, 7])
data3_input = []
csv_reader2 = csv.reader(open('data_question3.csv'))

for row in csv_reader2:
    data3_input.append(map(lambda x: int(x), row[0:7]))
print 1
for i in range(10):
    for j in range(7):
        data3[i][j] = data3_input[i][j]

return I, O, validation_I, validation_O, data3
```

```
#a, b, c, d , e= getlevel()
#print 1


def get_5_scale():
    csv_reader = csv.reader(open('data_RDNN.csv'))
    inner_input_array = []
    inner_output_array = []
    for row in csv_reader:
        if row[0] in common_inner:
            inner_input_array.append(map(lambda x: int(x), row[1:8]))
            inner_output_array.append(standarize4_inner(row[0]))

    input_x = np.zeros([1874, 7])
    for i in range(input_x.shape[0]):
        for x in range(input_x.shape[1]):
            input_x[i][x] = inner_input_array[i][x]

    I = np.zeros([90, 20, 7])
    count1 = -1
    for i in range(I.shape[0]):
        for x in range(I.shape[1]):
            count1 += 1
            for j in range(I.shape[2]):
                I[i][x][j] = input_x[count1][j]

    input_y = np.zeros([1874, 5])
    for i in range(input_y.shape[0]):
        for x in range(input_y.shape[1]):
            input_y[i][x] = inner_output_array[i][x]

    O = np.zeros([90, 20, 5])
    count2 = -1
    for i in range(O.shape[0]):
        for x in range(O.shape[1]):
```

```
                count2 += 1
                for j in range(O.shape[2]):
                    O[i][x][j] = input_y[count2][j]

    validation_I = np.zeros([100, 7])
    for i in range(100):
        for j in range(7):
            validation_I[i][j] = input_x[1000 + i][j]
    validation_O = np.zeros([100, 5])
    for i in range(100):
        for j in range(5):
            validation_O[i][j] = input_y[1000 + i][j]

    data3 = np.zeros([1, 7])
    data3_input = []
    csv_reader2 = csv.reader(open('data_question3.csv'))

    for row in csv_reader2:
        data3_input.append(map(lambda x: int(x), row[0:7]))
    print 1
    for j in range(7):
            data3[0][j] = data3_input[3][j]
    print data3

    return I, O, validation_I, validation_O, data3

print 1
a, b, c, d, e = get_5_scale()
print 1
```

**7. SVM_test.py**

```
#coding=utf-8
from __future__ import division
from sklearn import metrics
from sklearn import cross_validation
```

```
from sklearn.svm import LinearSVC
from sklearn.svm import *
from sklearn.multiclass import *

from sklearn.preprocessing import MultiLabelBinarizer
import numpy as np
from sklearn import metrics
from numpy import random
from read_csv_svm import *
X, Y , i, _= getdata2()


def multiclassSVM():
    count = 0
    model = OneVsOneClassifier(SVC(C=0.001)) #SVC 为二项分类器
    model.fit(X, Y)
    predicted = model.predict(i)
    # print predicted
    # print _
    for x in range(600):
        if (predicted[x] == _[x]):
            count += 1
    print count / 600

def multilabelSVM():
    count = 0
    model = OneVsOneClassifier(SVR(C = 1))
    model.fit(X, Y)#5000+数据放在一起训练
    # print X_test
    predicted = model.predict(i)
    # print predicted
    # print _
    for x in range(600):
        if (predicted[x] == _[x]):
            count += 1
```

```
    print count / 600

if __name__ == '__main__':
    multiclassSVM()
    # multilabelSVM()
```