Traditional Chinese Medicine (TCM) medical cases analyse

Lee Man Ho
Department of Computer Science
National Tsing Hua University
101, Section 2, Guangfu Road, Hsinchu City, Taiwan

Lin Chi Yu
Department of Computer Science
National Tsing Hua University
101, Section 2, Guangfu Road, Hsinchu City, Taiwan

Huang Shiu Peng
Department of Educational Psychology and Counseling
National Tsing Hua University
101, Section 2, Guangfu Road, Hsinchu City, Taiwan

Chiu Hsu Chen
Department of Kinesiology
National Tsing Hua University
101, Section 2, Guangfu Road, Hsinchu City, Taiwan

Wu Po Yi
Department of Computer Science
National Tsing Hua University
101, Section 2, Guangfu Road, Hsinchu City, Taiwan

Liu Chin Ting
Department of Computer Science
National Tsing Hua University
101, Section 2, Guangfu Road, Hsinchu City, Taiwan

***Abstract*—Traditional Chinese Medicine Generation using Neural Network and Support Vector Machine This paper presents a novel approach to generating traditional Chinese medicine (TCM) prescriptions using a Neural Network (NN) with multiple units, and Supports Vector Machine (SVM) algorithms. The objective is to leverage deep learning techniques for automating the formulation of herbal prescriptions based on established principles of traditional Chinese medicine. The study addresses the need for efficient and personalized prescription formulation by developing a computational model capable of capturing the intricate relationships between herbs and their therapeutic indications. Through a comprehensive analysis of the generated prescriptions, the research seeks to validate the model's adherence to traditional Chinese medicine principles and contribute to the integration of artificial intelligence in the field of herbal medicine.**

***Index Terms*—Traditional Chinese Medicine, Machine learning, Neural Network, Support Vector Machine**

I. Introduction

Traditional Chinese medicine (TCM) has a rich history of utilizing herbal prescriptions to treat a myriad of health conditions. As the demand for personalized and efficient healthcare solutions grows, there is a pressing need for innovative approaches to automate the formulation of TCM prescriptions. This study explores the application of neural network and support vector machine technologies to address this challenge. By harnessing the power of machine learning, we aim to develop a computational model capable of generating TCM prescriptions that align with established principles and demonstrate efficacy. This introduction provides an overview of the background and significance of the research, outlines the problem at hand, and introduces the methodology and objectives guiding our investigation.

II. Method

A. Experimental setup

1) Hardware: The program was executed on a workstation equipped with: intel core i7 13th gen (NN), intel core i7 8th (NN), intel core i5 10th gen (SVM).

2) Software: The neural network models were implemented using Python programming language and the TensorFlow deep learning framework. Additionally, the scikit-learn library was also employed at various stages of the program.

B. Data collection

We use the data provided by TA for our models. We once interviewed a TCM practitioner who explained that TCM has many different schools of thought, and each school has its own unique principles for prescribing medication. Therefore, the same symptoms presented to different TCM practitioners from different schools may result in completely different prescriptions. Our data mainly comes from the prescriptions given by Dr. Ni Haixia to different patients, so if a practitioner from a different school were to prescribe, the results could be completely different.

1) Data overview: Our raw data, stored in a CSV file, consists of individual rows representing medical cases, with each column providing different information about the case. This information could pertain to the patient's body status, symptoms, or prescribed medication, with some entries including dosage details. Most columns contain binary or categorical data, such as sleep quality, which can be classified into categories like "good", "normal", "lot of dreaming", or "feeling cold when sleep".

Notably, one column contains extensive text, seemingly comprising medical notes and diagnoses. This could include the doctor's comments, diagnoses, or the name of the prescription. However, the content varies across rows; for instance, some entries mention the prescription name, while others do not.

Our whole data contains about 800 rows of medical cases, and about 100 columns of information.

C. Problem define

We define our problem as binary classifiers of each medicine, in which we wish to predict whether a medicine should be prescribed or not according to the features of the medical case, in other word, we define the features of the study as symptoms and body status of the medical case, and labels as medicine prescribed.

D. Data preprocessing

1) Features: To start with a simple model, we convert all the data related to symptoms and body status to mostly binary data, for example, if the data mention the patient sleep badly, we convert to a column indicating "sleep badly" and set the value to "1". If a feature is not mentioned, we treat it as "0". Some data are treated as categorical where we grouped features related to the same field to different categories under the same feature.

2) Labels: As dosage of medicine is missing or for many medical cases, we convert all medicine into binary data, where 1 indicates presence, 0 indicates absence.

3) Data filtering: As mentioned above, one of the columns contains long text, which appear to be malicious notes and diagnosis about the medical case, however, since the format and content of this column highly differ for each instance, it is hard to convert the information into numerical representation, therefore this text-based data is filtered out.

In light of these discoveries, we implemented a threshold mechanism, termed "DeleteMedThreshold," into our code. This mechanism selectively excludes medications that occur below a specified threshold, enabling us to concentrate on the more commonly utilized medications within the dataset. Considering the sparse nature of the data, we expect that eliminating medications with infrequent occurrences will improve the overall performance of the model.

4) Comparison: The 2 data preprocessing procedures are basically identical for both models, they both convert data to mostly binary data, delete medications with less occurrence, etc. However, for the SVM model, we convert more columns into numerical presentation.

Dense	Units(nodes)	Activation
1st	16	relu
2nd	32	relu
3rd	32	relu
4th	2	softmax

TABLE I
INITIAL MODEL STRUCTURE

Maximum epoch	100
early stop	None
Optimizer	Adam
Learning rate	0.01
Loss function	$\text{sparse}_c ategorical_c orssentropy$

TABLE II
INITIAL MODEL HYPERPARAMETERS

E. Models

We decided to use NN and SVM as our main algorithm. Since other previous studies indicate that SVM and NN perform effectively in disease diagnosis and disease treatment. We develop different machine learning models for NN and SVM to simulate how a trained Chinese medicine professional would prescribe under real-world scenarios.

F. Algorithms and methods: NN

1) Initial Model: Initially, we considered two approaches:

1. viewing all medications as a collective response, one model that predict all medicine will be trained;
2. training distinct models for different medications

After further comparison, we found out that models trained for individual medication performed better, consequently, we shifted our focus towards training separate models for each medication.

The neural network model architecture is constructed using the sequential model of Keras library (keras.model.sequential) In order to train a distinct model for each medication, we designed each model as a binary classifier for the respective medication. All models were trained with identical specifications in a given trial. We selected "sparse_categorical_crossentropy" as our loss function, a choice supported by research as an effective starting point for binary classifiers. These models predict the probabilities of the medication being 0 and 1, in which the class of higher probability indicates the model's prediction. As a starting point, we constructed a simple initial model comprising three dense layers.

We split the data into training and validation set to a ratio of 2:8.

Specification of the Initial Model is shown in table 1 and table 2

2) Problems with initial model: To evaluate the training result, we calculated the mean accuracy, precision, recall and f1 score of the models of each medicine, and compared the training and validation set.

We found that the accuracy of both train set and validation set is high, however, the precision, recall and f1 score of both training set and validation set are low, which indicates that the model is not well trained (table 3).

	Training set	Validation set
Accuracy	0.9103	0.8856
Precision	0.6758	0.5218
Recall	0.3193	0.2212
F1 score	0.4337	0.3107

TABLE III
INITIAL MODEL EVALUATION

Dense	Units(nodes)	Activation
1st	32	relu
2nd	64	relu
3rd	128	relu
4th	64	relu
5th	32	relu
6th	2	softmax

TABLE IV
SECOND MODEL ARCHITECTURE

We further analyzed the confusion matrix of the model, and found that the model is biased towards predicting the majority class, which is 0 (no medicine prescribed).

We further analyzed the data and found that the data is highly imbalanced, in particular,the number of positive samples (the medicine is prescribed) is much less than the number of negative samples (the medicine is not prescribed).

When looking into the prediction of instances of each medicine, we found that the model predict most cases to 0, for some medicine, it even predicted all cases to 0.

Due to the highly imbalanced data, although the accuracy is high, but this does not indicate that the model is well trained.

Since our initial model is relatively simple, with only 3 dense layers and trained for 100 epochs, we first assume that the model is underfitted, so we tried to increase the complexity of the model and train for more epochs.

3) Second Model: In order to increase complexity of the initial model, our second model is designed as shown in table 4 and in table 5 :

We found that the accuracy of the training set is higher than the initial model, however, the accuracy of the validation set is not significantly higher than the initial model, which suggests that the model might be overfitted. The result of the second model can be found in table 6.

To experiment with various hyperparameters and strategies for enhancing model performance, we developed a Jupyter Notebook that enables flexible adjustment of model specifications.

Our experimental design involved tuning various parameters such as the number of dense layers, the number of neurons, the activation function, the number of epochs, and the patience for early stopping.

Maximum epoch	1000
Early stop	None
Optimizer	Adam
Learning rate	0.001
Loss function	$\text{Sparse}_{c}ategorical_{c}orssentropy$

TABLE V
SECOND MODEL HYPERPARAMETERS

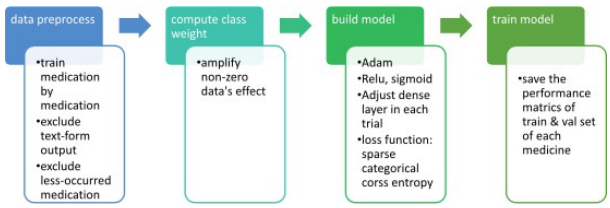

Fig. 1. Steps of experiment

	Training set	Validation set
Accuracy	0.925	0.8765
precision	0.7565	0.4472
Recall	0.4516	0.8765
F1 score	0.5656	0.3235

TABLE VI
RESULT OF SECOND MODEL

We implemented and tested several strategies to assess their impact on model performance. These included adding a dropout layer, incorporating class weights, modifying the loss function, and filtering out infrequently appearing medications. We will discuss the details of these strategies in the subsequent section. The steps of our experiment are illustrated in figure 1.

4) Striving Balance between Overfitting and Underfitting : To prevent the model from overfitting, we tried to add dropout layers to the model, also, we tried to add an early stop to the model. If the validation loss does not decrease for certain epochs, the training will stop.

To strive a balance between overfitting and underfitting, we adjusted the number of layers and number of neurons in each layer.

5) Tackle Imbalanced Data: Upon conducting a detailed investigation of the initial and second models of each medication, we discovered that many of the models predicted all cases to 0 (meaning the medicine should not be prescribed), with only a few exceptions.

Detailed result please refer to the table in the appendix.

We plotted the occurrence of '1' in the dataset against the F1 score from the validation set for each medication, and found that there seems to be a positive correlation between the two variables. However, there are numerous instances where, despite a high occurrence of a particular medication, the F1 score remains at zero (figure 2). Model specification: 2-64-128-64-32 units, activation: relu, optimizer: Adam, learning rate: 0.001, epochs: 1000

6) Implementing a Threshold Mechanism : In response to these findings, we introduced a threshold mechanism, "DeleteMedThreshold", in our code. This mechanism filters out medications with occurrences below a defined threshold.

This strategy allows us to focus on the most frequently used medications in the dataset. Given the data's sparsity, we anticipate that removing medications with low occurrences will enhance the model's performance.

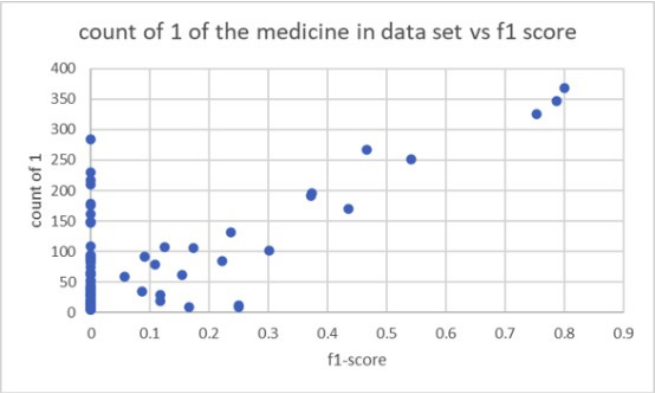

Fig. 2. count of 1 vs. f1 score

7) Balancing Classes with Weights: To further improve our model, we attempted to balance the number of positive and negative samples in the dataset by adding class weights.

We devised a custom function to assign these weights. Specifically, the weight for class 0 is the ratio of class 1 frequency to total frequency, and vice versa (inversely proportional).

8) Modifying the Loss Function: We re-framed the problem to predict the probability of occurrence for each medication, subsequently adopting binary cross-entropy as our loss function.

9) Feature Selection: We refined our feature selection to focus solely on symptoms, disregarding features related to body status, based on our hypothesis that symptoms could have a more direct correlation with prescribed medications, thereby potentially enhancing the predictive accuracy of our model.

10) Adding Dropout layers: We also tried adding dropout layers. Dropout is a regularization technique that prevents overfitting by randomly setting a fraction of input units to 0 at each update during training time.

11) Evaluating Various Strategies: We assessed the model's performance under different strategies and compared the results. Our findings revealed that filtering out infrequently appearing medications resulted in the most significant improvement in the model's performance.

However, other strategies such as adding class weights, incorporating dropout layers, modifying the loss function, and using only symptoms as features did not yield any substantial improvements in performance.

12) Final Model: After experimenting with different hyperparameters and strategies, we found that the following set of hyperparameters and strategies yield the best performance (NN final model), and it is shown in table 7.

G. Algorithms and methods: SVM

1) Initial Model: Initially, we considered two approaches:

1) viewing all medications as a collective response, one model that predicts all medicines will be trained.
2) training distinct models for different medications

Maximum Epoch	2000
Early Stop	Yes
Optimizer	Adam
Learning rate	0.005
Loss Function	Sigmoid

TABLE VII
FINAL MODEL 1

Dense	Units(Nodes)	Activation
1st	64	Sigmoid
F1 score		
0.3299		

TABLE VIII
NN MODEL RESULT 1

After further comparison, we found out that we found that the result is totally the same.

A Support Vector Machine (SVM) model is constructed using the scikit-learn library. When viewing all medications as a collective response, the SVM model is configured as a MultiOutputClassifier to handle the multi-label nature of the prescription data. The RBF (Radial Basis Function) kernel is chosen with a regularization parameter (C) set to 5.0.

Key hyperparameters include the choice of using RBF kernel, and the setting of the regularization parameter, which, in this case, is set to 5.0. It controls the trade-off between having a smooth decision boundary and correctly classifying the training points correctly.

The threshold mechanism, "DeleteMedThreshold" is also use in this model to filter out medications with occurrences below a defined threshold.

2) Problems with the initial model: Just like what we do in the NN model, we calculated the mean accuracy, f1 score of the models of each medicine, and compared the training and validation set. Though the result is not bad, we still want to see whether adding weights to different classes will result in a better result or not.

3) Final Model: The key hyperparameters are totally the same as the first model, the only difference is that we add weight to different classes. The way we add weight is the same as what we do in the NN model, the weight for class 0 is the ratio of class 1 frequency to total frequency, and vice versa. However, there's not much improvement after adding weights.

III. RESULTS

A. Results of NN model

1) All the models' results (training data's f1-score): The results of the model, which implements an occurrence threshold to filter out medicines with a count of 0 and does not utilize weights (NN model result 1), are shown in table 8.

The results of the model, which implements an occurrence threshold to filter out medicines with a count of 0 and utilizes weights (NN model result 2), are shown in table 9.

The results of the model, which implements an occurrence threshold to filter out medicines with a count of 100 and does

Dense	Units(Nodes)	Activation
1st	64	Sigmoid
F1-score		
0.3233		

TABLE IX
NN MODEL RESULT 2

Dense	Units(Nodes)	Activation
1st	64	Sigmoid
F1-score		
0.4166		

TABLE X
NN MODEL RESULT 3

not utilize weights (NN model result 3), are shown in table 10.

The results of the model, which implements an occurrence threshold to filter out medicines with a count of 100 and utilizes weights (NN model result 4), are shown in table 11.

The results of the model, which implements an occurrence threshold to filter out medicines with a count of 250 and does not utilize weights (NN model result 5), are shown in table 12.

The results of the model, which implements an occurrence threshold to filter out medicines with a count of 250 and utilizes weights (NN model result 6) (yields best performance), are shown in table 13.

The model's F1-score performances without utilizing weights can be found in table 14

The model's F1-score performances utilizing weights can be found in table 15

2) Results of the final model: The results of the model, which implements an occurrence threshold to filter out medicines with a count of 0 and does not utilize weights (NN final model result 1), are shown in table 16.

The results of the model, which implements an occurrence threshold to filter out medicines with a count of 100 and utilizes weights (NN final model result 2), are shown in table 17.

The results of the model, which implements an occurrence threshold to filter out medicines with a count of 250 and does not utilize weights (NN final model result 3), are shown in table 18.

Dense	Units(Nodes)	Activation
1st	64	Sigmoid
F1-score		
0.4308		

TABLE XI
NN MODEL RESULT 4

Dense	Units(Nodes)	Activation
1st	32	Sigmoid
2nd	16	Sigmoid
F1-score		
0.5384		

TABLE XII
NN MODEL RESULT 5

Dense	Units(Nodes)	Activation
1st	32	Sigmoid
F1-score		
0.5318		

TABLE XIII
NN MODEL RESULT 6

Dense/Thresholds	0	100	250
16	0.3153	0.4058	0.4904
32	0.3095	0.3841	0.5181
32-16	0.3237	0.3787	0.5383
64	0.3299	0.4165	0.5294
64-16	0.3177	0.3793	0.4889
64-32	0.32	0.4049	0.5208
64-32-16	0.313	0.3879	0.4422

TABLE XIV
PERFORMANCES WITHOUT USING WEIGHTS

B. Results of SVM model

1) Results of the initial model: The results of the model, which implements an occurrence threshold to filter out medicines with a count of 0 and does not utilize weights (SVM initial model result 1), are shown in table 19.

The results of the model, which implements an occurrence threshold to filter out medicines with a count of 100 and does not utilize weights (SVM initial model result 2), are shown in table 20.

The results of the model, which implements an occurrence threshold to filter out medicines with a count of 250 and does not utilize weights (SVM initial model result 3), are shown in table 21.

2) Results of the second model: The results of the model, which implements an occurrence threshold to filter out medicines with a count of 0 and utilizes weights (SVM second model result 1), are shown in table 22.

The results of the model, which implements an occurrence threshold to filter out medicines with a count of 100 and utilizes weights (SVM second model result 2), are shown in table 23.

The results of the model, which implements an occurrence threshold to filter out medicines with a count of 250 and

Dense/Thresholds	0	100	250
16	0.2985	0.3955	0.52
32	0.2488	0.4049	0.5317
32-16	0.2924	0.4206	0.4836
64	0.3232	0.4308	0.5155
64-16	0.2598	0.3717	0.5258
64-32	0.2611	0.4053	0.5263
64-32-16	0.2832	0.3989	0.5191

TABLE XV
PERFORMANCES UTILIZING WEIGHTS

	Training set	Validation set
Precision	0.992	0.366
Recall	0.955	0.300
F1-score	0.973	0.330

TABLE XVI
NN FINAL MODEL RESULT 1

	Training set	Validation set
Precision	0.961	0.463
Recall	0.97	0.403
F1-score	0.966	0.431

TABLE XVII
NN FINAL MODEL RESULT 2

	Training set	Validation set
Precision	0.981	0.559
Recall	0.97	0.519
F1-score	0.976	0.538

TABLE XVIII
NN FINAL MODEL RESULT 3

	Training Set	Validation Set
Precision	0.9881	0.6526
Recall	0.9377	0.5545
F1 score	0.9619	0.5933

TABLE XXI
SVM INITIAL MODEL RESULT 3

	Training Set	Validation Set
Precision	0.7677	0.3363
Recall	0.9893	0.3085
F1 score	0.8501	0.3025

TABLE XXII
SVM SECOND MODEL RESULT 1

utilizes weights (SVM second model result 3), are shown in table 24.

IV. DISCUSSION / CONCLUSION

A. Imbalanced Data

Our dataset comprises patient characteristics and prescriptions. Each patient's characteristics represent only a fraction of all possible features, and likewise, each medication within the prescriptions represents only a fraction of all medications present in the dataset. Consequently, this makes our data set sparse, both the labels and features in the dataset are highly imbalanced, with a significantly larger number of occurrences of 0s compared to 1s.

Many medicinal herbs are used so infrequently that it results in a predicted outcome of 0. Therefore, if we predict all as 0, the accuracy of the prediction will be high. However, this does not mean that our model is accurate. We hope to predict 1 even if it appears infrequently, so we put weight on non-zero data to make them significant

However, after adding weights, the accuracy of our model did not improve. We believe the possible reason is that we did not find the most suitable weights, which led to predictions that should have been 0 becoming 1, and the number of incorrect predictions is greater than the number of predictions that should have been 1.

B. The reason 250 threshold's model has better result

Less frequently occurring medicines may exhibit collinearity with commonly occurring medicines, meaning that the effects of these less frequent medicines are similar to those of commonly used medicines. For example, different doctors may prescribe different medicines for the same condition, or a primary medicine may be used during the first visit and an additional medicine may be added during a subsequent visit for the same condition. By filtering out less frequently occurring medicines, when evaluating the same condition, the model can select a single medicine, resulting in higher accuracy.

C. Low F1 Score in Validation Set in NN Model

For some medicines, one of the reasons is because there are very few positive values in the validation set. For example, in one of the rounds of training, the medicine "Shengma" only occurs twice in 159 instances in the validation set. The trained model missed the 2 positive cases, resulting in 0 F1 score. In this cases where positive cases are so few in the validation set ,it is difficult to fairly evaluate the performance of the trained model.

D. Comparison of Performance of SVM and NN

SVM in generating traditional Chinese medicine prescriptions. The model consistently produced prescriptions that align with established principles of TCM, as evidenced by the high F1-scores. We observed that the result of our SVM model is better than our NN model. We think it's because our dataset is too small, so we didn't reach the point where the performance of NN will surpass the performance of SVM, just like what is shown in the figure below. We believe that once we get a dataset that is big enough, the performance of NN will become better. While our findings are promising, it's crucial to acknowledge certain limitations. The dataset's

	Training set	validation set
Precision	0.9980	0.4058
Recall	0.9089	0.2329
F1-score	0.9499	0.2732

TABLE XIX
SVM INITIAL MODEL RESULT 1

	Training Set	Validation Set
Precision	0.9941	0.5734
Recall	0.9189	0.3999
F1 score	0.9546	0.4577

TABLE XX
SVM INITIAL MODEL RESULT 2

	Training Set	Validation Set
Precision	0.9214	0.5516
Recall	0.9756	0.4804
F1 score	0.9469	0.5068

TABLE XXIII
SVM SECOND MODEL RESULT 2

	Training Set	Validation Set
Precision	0.9693	0.6458
Recall	0.9636	0.5816
F1 score	0.9663	0.6084

TABLE XXIV
SVM SECOND MODEL RESULT 3

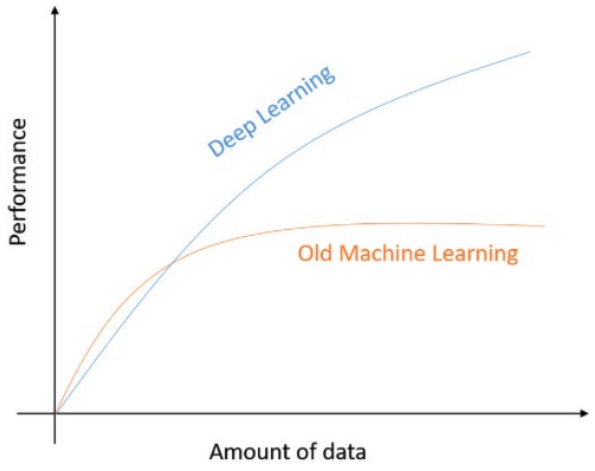

Fig. 3. Deep learning v.s Old Machine Learning

representativeness and the generalizability of the model to diverse patient populations should be considered. Additionally, alternative explanations for certain prescription patterns should be explored in future studies.

E. Generalizability of the Model

The practical implications of our findings extend to personalized medicine and computer-assisted prescription formulation. By automating the generation of TCM prescriptions, our model has the potential to assist practitioners in developing tailored treatment plans efficiently. Nevertheless, we should take heed of the model's performance in real-world scenarios, as we mentioned above, this model is specifically for Dr. Ni Haixia's school of treatment, so it might not be applicable for all TCM prescriptions. Additionally, we believed that the F1-score would be better if the data is more balanced (more balanced 0s & 1s), however, in real-world scenarios, it is impractical to assume that we can acquire a balanced data set to train, considering the abundant variety of herbal medicines. Future research should explore the integration of patient-specific data and real-world clinical outcomes to further validate the model's applicability.

In conclusion, our study contributes to the growing intersection of artificial intelligence and traditional medicine, offering a viable and effective approach to TCM prescription generation. The limitations highlighted underscore the need for ongoing refinement and validation, paving the way for enhanced healthcare solutions grounded in the principles of traditional Chinese medicine.

V. Data and Code Availability

link to our Github repository for NN model: https://github.com/110062122domingo/TCM_NN
link to our Github repository for SVM model: https://github.com/guavaaa0826/ML_TCM_SVM
Noted that the data we use is also included in our repository

VI. Author Contribution Statements

W.H.L. (19%): NN model developer
P.Y.W. (19%): SVM model developer
C.Y.L. (17%): Tester, Designing model architecture
H.C.C. (15%): Report designer, Data organizer
S.P.H. (15%): Report designer, Presenter
C.T.L. (15%): Project manager, Visualizing data

References

[1] Z. Liu et al., "A novel transfer learning model for traditional herbal medicine prescription generation from unstructured resources and knowledge," *Artificial Intelligence in Medicine*, vol. 124, 2022.
[2] Zhao, Changbo, et al. "Advances in patient classification for traditional Chinese medicine: a machine learning perspective." Evidence-based complementary and alternative medicine: eCAM 2015 (2015).
[3] Ma, Suya, et al. "Machine learning in TCM with natural products and molecules: current status and future perspectives." Chinese Medicine 18.1 (2023): 1-17.
[4] ctext.org. Shang Han Lun. ctext.org. https://ctext.org/shang-han-lun/zh.
[5] ctext.org. (n.d.). Wiki page. ctext.org. https://ctext.org/wiki.pl?if=gb&res=763639
[6] Box User, "Research Data on Traditional Chinese Medicine," Box, 2022. [Online]. Available: https://app.box.com/s/i9z8dy7tdz2f69l66427577jnah0mym8/folder/149622394353