Disease Predictor — Symptom-based Big Data Analysis

VM455

Jiajin WU, 517370910229

Sheng CEN, 517370910045

Fall 2020

12/07/2020

Executive Summary

Combination of different symptoms for patients may serve as an indicator of certain diseases. In this project, we apply supervised learning to dig into many patients' cases of disease, as well as their reported symptoms, and dedicated to finding a mechanism to predict a given patient's disease he/she mostly likely have caught as accurate as possible, given his/her reported combination of symptoms. Then we transform it into a classification problem, using SVM, and decision tree (w/o ensemble methods), and compare their performances with regards to training speed and accuracy. Dataset is pre-processed first to eliminate the effect of multi-collinearity and discreteness. In order to achieve high training efficiency, cross validation is applied to discover the effect of different fold number k.

Keywords:

Support vector machines; Principle component analysis; Decision tree; Ensemble learning; Cross validation; Correlation; Disease prediction

1 Background and Motivation

With population surging in large cities like Shanghai, patients have to wait in long queues to wait for diagnosis by doctors. Thus, it becomes necessary to design a machine-learning-based predictor to automatically offer pre-diagnosis to each potential patient given his/her reported combinations of symptoms.

Fig. 1 gives us a picture in mind that concurrence of various symptoms may determine certain diseases. In the figure, Symptom S2 and Sm links Disease D4 with bold lines, indicating that the combination of S2 and Sm may lead to D4 in large probability. We believe that given large data set of patients' information on their symptoms and diseases, we could find a pattern to predict some other patient's possible disease given his/her symptoms.

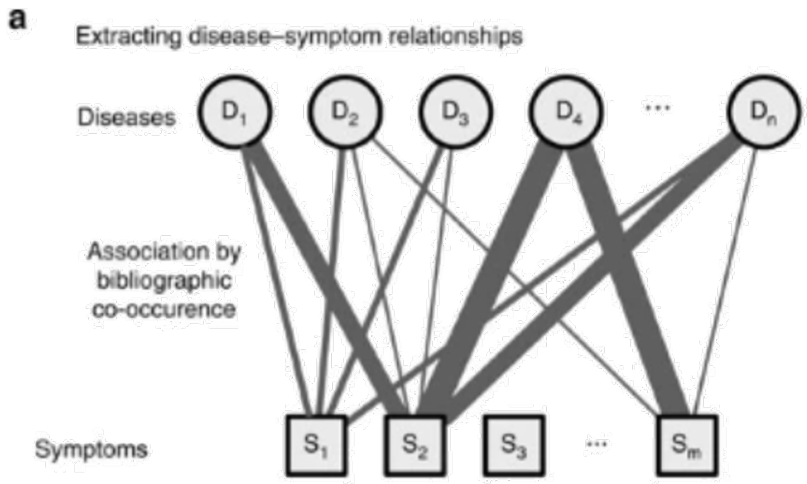


Figure 1: Extracting the disease–symptom relationships from PubMed bibliographic literature database.

2 Project Objectives and Novelty

The project is categorized into data-driven big data analysis. It employs a large data set containing many patients information on their reported symptom(s) and their diagnosed diseases. Our main task is to find some methods to train the data in a scientific manner, and predict certain patients' disease as accurate as possible. The prediction is expected to give the most possible disease one caught given certain combination of symptoms. Therefore, the prediction is based on supervised learning in classification.

Novelty in the project includes:

1. Use correlation matrix to find correlation between various features (symptoms) to eliminate multi-collinearity. Results would be affected if there is strong correlation between two features.

2. Compare prediction accuracy using different training methods: SVM, decision trees (w/o PCA, bagging, boosting), and make necessary analysis. [3].

3. Apply cross-validation to increase training accuracy, and study the relationship between the number of folders and accuracy.

3 Literature Review

We have searched for some advanced methods on data-driven analysis, such as feature correlation analysis, improvement of decision tree and SVM, etc., which gives us some insight on what others have researched in the field of data pre-processing and model construction part.

1. **"Correlation Analysis to Identify the Effective Data in Machine Learning: Prediction of Depressive Disorder and Emotion States"**

This paper [5] introduces a statistical method on how to find a proper feature set (among many features) in machine learning problems. The selected features should have low correlations among them, as well as serve as strong indicator for the dependent variable.

The basic formula of depicting correlation is Pearson's correlation coefficient: $C_{A,B} = \frac{\text{Covariance}(A,B)}{\sigma_A \sigma_B}$

Then, using this formula, we could define a "good" feature set as one that has low correlations among them, and has large correlations between these features and the dependent variable. So the concept of "merit" is introduced:

$$\text{Merit} = \frac{\overline{kavg\,(\text{corr}_{fc})}}{\sqrt{k + k(k-1)\,\text{avg}\,(\text{corr}_{ff})}}$$

where "$corr_{fc}$" is the correlation between a feature and the target dependent variable, "$corr_{ff}$" is the correlation between two features. The upper line here denotes the average value for all features in the feature set. "k" means the feature number in the feature set. We could rank each candidate feature set in terms of decreasing "merit" value.

The method can be applied in which lots of correlations exist among

features, and there are lots of features (i.e. high dimension data), that there exist apparent redundant issues.

Pros:

(a) Aggregate both the metric of correlations among features, as well as whether these features are strong indicators for the dependent variables (significance of fitting coefficients).

(b) A scientific way for coping with high-dimensional data.

Cons:

(a) Highly computed, since we need to identify each possible set.

(b) It does not make sense for small data set.

2. **"The Max-Cut Decision Tree: Improving on the Accuracy and Running Time of Decision Trees"**

This paper [1] improves the traditional decision tree splitting method using Gini Impurity. The traditional way is using Gini Impurity formula at each split:

$$\sum_{c} p_c \left(1 - p_c\right)$$

p_c is the proportion of samples for class c, and the larger its value, the more impure of the class distribution. So our goal is to minimize the average Gini Impurity value for the two splitting subsets. But the

method does not take into account the distance of the split, just the partition induced by the split.

So the proposed optimization method is to "find a threshold that seeks to maximize the sum of the weighted arcs between the two subsets of the partition induced by that threshold, where the arcs' weights are determined by the distance in the specified feature of the two nodes if those nodes are from different classes and zero otherwise."

$$\max_{\theta} \sum_{\{i|x_{i,j}\leq\theta\}\{k|x_{k,j}>\theta\}} 1_{y_i \neq y_k} |x_{i,j} - x_{k,j}|$$

in which θ is the threshold for certain feature, y is the belonging class, and x denotes the sample.

To speed up the computation such that not each feature is evaluated, we could use PCA to reduce dimension at each tree split. There are two kinds of solutions:

1) Node Features PCA: At each tree split, use the remaining samples to perform PCA.

2) Node Means PCA: At each tree split, locally calculating the mean position of each of the possible 'rest' collections, to feature difference between classes.

In the paper, the authors choose the second way. This method can be applied to all solutions which involves decision trees, since over-fitting

is always a problem people want to overcome. And the selection of decision argument is fast-implemented due to introduction of PCA.

Pros:

1) Significantly reduce the computational cost for implementing decision tree. 2) Group heterogeneity is enlarged for each decision node, not prone to over-fitting.

Cons:

1) Only applied to binary classification for each decision node 2) Hard to interpret in terms of principle components

3. **"A novel robust kernel for classifying high-dimensional data using Support Vector Machines"**

This paper [2] proposes a novel kernel for SVM, applicable for high-dimensional data. It uses the idea of co-similarity, to find latent relationships between documents. The task is that we need to categorize certain document according to its word distribution. We need to find the similarity matrix for different documents and words respectively, denoted as R and C:

The similarity score interpreted in Figure 2:

For document d2 and d3, they have only w4 in common, denoted as path-1 relations. If we consider d2 and d5, they have no direct word

$$R_{ij}^{(k)} = \begin{cases} 1 & if\ i = j \\ \left[\boldsymbol{X}\boldsymbol{C}^{(k-1)}\boldsymbol{X}^T\right) \circledast \boldsymbol{N}\boldsymbol{R}\right]_{ij} & otherwise \end{cases} \quad (22)$$

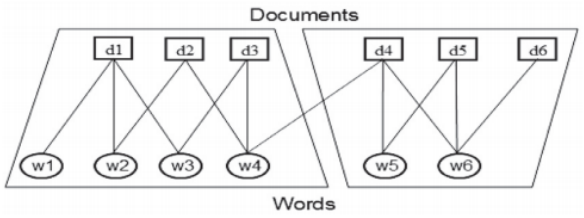
$$C_{ij}^{(k)} = \begin{cases} 1 & if\ i = j \\ \left[\left(\boldsymbol{X}^T\boldsymbol{R}^{(k-1)}\boldsymbol{X}\right) \circledast \boldsymbol{N}\boldsymbol{C}\right]_{ij} & otherwise \end{cases} \quad (23)$$

where '⊛' denotes the Hadamard multiplication (i.e., $A_{ik} = B_{ik} * C_{ik}$) and **X** is the data matrix.

Figure 2: R, C formula.

	w_1	w_2	w_3	w_4	w_5	w_6
d_1	1	1	1	0	0	0
d_2	0	1	0	1	0	0
d_3	0	0	1	1	0	0
d_4	0	0	0	1	1	1
d_5	0	0	0	0	1	1
d_6	0	0	0	0	0	1

(a)



(b)

Figure 3: Similarity score intepretation.

in common, but can be linked by d2-w4-d4-w6-d5 path, more indirect relation.

Since words that mostly occur in documents of the same category should have higher similarity values (hence, paths), we also add additional similarity score for documents falling in the same category.

Pros:

1) suitable for sparse, dyadic data where direct co-occurrences are not necessary common as in the case of textual data, link-analysis in social media networks, co-authorship, etc.

2) Not sensitive to noisy data.

3) Computation is not so intense.

Cons:

Not applicable for collection of data about the same samples but from different sources and represented by different feature set.

4. **"Approximate Cross-Validation in High Dimensions with Guarantees"**

This paper [7] researches the performance of Leave-one-out cross validation (LOOCV), which is a kind of cross validation methods, in high dimensions. It is widely acknowledged that although the LOOCV works well for small, fixed dimensions, the accuracy deteriorates dramatically in high dimensions. However, the research team claims that in the case when the high-dimensional parameters are sparse, the approximation of LOOCV can also perform well both empirically and theoretically. Therefore, they argue that we can reduce the problem to work within an empirically recovered (small) support, which is straightforward to implement.

This can be applied in the cross validation of high-dimensional supervised learning to optimize the trained model and test it on other testing sets. Usually, the dummy matrix is sparse everywhere if the data has a lot of categories, and since the LOOCV is pretty fast and accurate, the running time could be reduced and the efficiency of machine learning will increase.

We can use this kind of cross validation to train and test our dataset because our dataset is exactly high-dimensional, sparse matrix. This method could save a lot of time.

Pros:

Suitable for sparse datasets and can run fast and accurate for the cross-validation procedure.

Cons:

The method is only an approximation for the LOOCV, and it is no suitable for high-dimensional data sets that are not sparse.

5. **"A general model for fuzzy decision tree and fuzzy random forest"**

This paper [8] researches the method to solve the problems of risk classification and prediction. One method is fuzzy logic, which is used to deal with the ambiguous data, which is suitable for the classification and prediction. However, fuzzy logic depends on the characteristics of data and objectives, so the authors propose a general membership function model for fuzzy sets (GMFMFS) in fuzzy decision tree and extend it to the fuzzy random forest. Also, they have proved that the method could be extended to a non-linear membership functions without a significant increase in computing complexity.

This newly introduced model could be programmed as fussy decision trees or fussy random forest algorithms to be applied to some fussy

data in our daily life, like the US credit, Susy dataset of UCI, ad synthetic datasets of big data. This solves the limitation of fussy logic and therefore more fussy sets in the real life can be utilized.

We will not use this method because it requires us to program everything from the theory by ourselves, which is beyond our capability.

Pros:

To deal with the fussy dataset in the real world and apply the decision tree method to train a usable model

Cons:

Requires strong mathematical background to implement the new algorithms.

4 The Raw Data Set

The raw data set contains about 4950 patient cases with 50 kinds of diseases described by 132 kinds of symptoms. For each kind of diseases, the number of symptoms varies from 1 to 18 or so, so we have converted the raw data set with random number of valid columns into a dummy 4950*132 matrix with only 0 and 1. Zero means that this kind of symptom is not recorded in this case for this kind of disease, and one represents just the opposite meaning.

5 Methods

The method we have used is supervised machine learning. Our focus is on the classification part because our goal is to implement a feasible disease predictor.

We first apply the correlation matrix to pre-process the data, in order to eliminate correlation among features (otherwise, multicollinearity would happen). And the operation would also transform discrete feature to continuous one, which is more meaningful. Also, PCA serves as an optional operation to reduce data dimension, more convenient for further processing.

Then due to the characteristic of this problem, i.e. supervised learning & classification, we could apply SVM and decision tree (compare the effect of ensemble learning) to study the accuracy and training speed of the respective algorithm. In order to train the data more scientifically, we apply cross-validation (with parameter, i.e. the number of folds, tuning) to achieve more reasonable effect. We could then study whether a large folds would lead to a more accurate prediction.

6 Results and Discussion

6.1 Comparison of Different Algorithms

6.1.1 Support Vector Machine (SVM)

According to the characteristic (high dimension, discrete, relatively small size) of our data set, we suppose that the Support Vector Machine (SVM) method is an appropriate supervised learning method to train the model. For

each disease, the symptoms should be very different (in many dimensions) and should be separated by a limited number of hyper-planes easily. The kernel function that we have chosen is linear, and the kernel scale is one by default. Although it takes quite a long time (83 seconds, compared to the decision tree: less than 2 seconds) to train the high dimensional SVM model, the accuracy is astonishingly high: 100%. Even for the simplest SVM method with linear kernel function, such accuracy proves that it is the best training method for the prognosis system.

6.1.2 Decision Trees with Different Splitting Numbers

To train a classifier, it is always acceptable to use the decision tree method because making judgements according to certain criteria is the most intuitive way to distinguish two categories. The most important issue in this process is to decide the criteria and the maximum splitting number.

The maximum splitting number refers to the number of steps of judgements. If we have set the maximum splitting number as 30, it means that the model should not take more than 30 steps to make judgments and find the result. It is trivial that more steps of judgement (more splitting branches of the decision tree) adds to the accuracy of the classifier, but the risk of being over fitted also increases. Therefore, the maximum splitting number of the decision tree should be chosen carefully.

To further research the relationship between splitting number and accuracy of the model, we make a plot as follows (Fig. 4). Surprisingly, we have found that the accuracy is perfectly proportional to the maximum splitting number, ranging from 12% to 100%. The training time is generally fast,

varying from 0.65 seconds to 2 seconds as the increase of splitting numbers. This means that decision tree method is flexible to use and adjust depending on the training speed and accuracy required by the specifications.

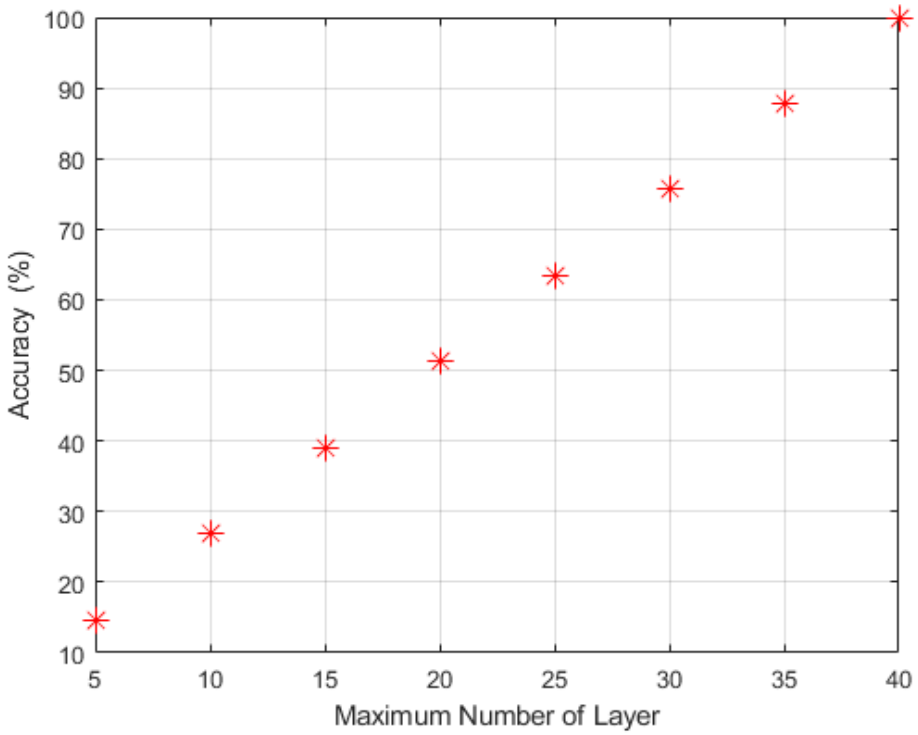


Figure 4: The plot of accuracy versus maximum number of layers.

6.1.3 Boosting Tree and Bagging Tree

The boosting and bagging trees are actually the refinement of decision tree model in the aspect of sampling and re-training. We have applied both methods to train our data set and found that the performance of boosting tree is much better than the bagging tree (99.7%, 16 seconds v.s. 49.1%, 7 seconds). This may due to the fact that for training sets with plenty of features and not very large in size, combining multiple classifiers to train the

whole data set is more efficient than splitting the data set into small sections and train them separately with one classifier.

6.2 Data Pre-processing with Correlation Matrix

In reality, it is common for multiple symptoms to appear together. For example, when you have the symptom of high fever, it is possible that you may also have the symptom of vomit because both them are closely related to your immune system against virus. Based on this fact, we think that there might be some correlation between symptoms naturally. Then, the high fever (named as Symptom 1) and vomit (named as Symptom 2) can be defined as "correlated" with a correlation coefficient c_{12}. Apparently, $c_{12} = c_{21}$ and $c_{nn} = 1$ for all integers. If we apply this definition to the rest of symptoms, we can build a "correlation matrix" that describes the correlation between each pair of symptoms. Assume that there are n kinds of different symptoms, the complete symmetric matrix of correlation should be:

$$C_{n\times n} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix}$$

This can help us pre-process our raw data set which are possibly incomplete. Considering the fact that most of the patients have not received professional medical training, their reports of the symptoms may have missing or wrong symptoms. With the help of the correlation matrix between symp-

toms, our target is to restore the missing or error entities in the data set appropriately.

Here is our method: now we have the correlation matrix $C_{n\times n}$ and the raw data set $A_{m\times n}$, representing that the training set has m cases. Each of the cases may have $(a = 1)$ or have not $(a = 0)$ each of those n symptoms. The complete training matrix of m cases and n symptoms should be:

$$A_{m\times n} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Then, we multiply the two matrix together to get the new training matrix. Notice that the rule of matrix multiplication requires the equal columns and rows of the matrices, so we take the transpose to ensure the consistence of dimension.

$$A'_{m\times n} = \left(C_{n\times n} * A^T_{m\times n}\right)^T$$

To calculate element by element, the adjusted item a'_{ij} in the new matrix $A'_{m\times n}$ equals to:

$$a'_{ij} = \left(C_{n\times n} * A^T_{m\times n}\right)_{ji} = \sum_{k=1}^{n} c_{jk}a_{ik} = \sum_{k=1}^{n} a_{ik}c_{kj}$$

Let us interpret the equation: the adjusted value of Symptom j in the case i equals to the summation of the original values of all symptoms multiplied by their correlation coefficient with respect to Symptom j. Generally speaking, When we are looking at the essence of a certain symptom, we also take

the essence of all other symptoms into consideration to determine the "generalized" essence of that symptom. This interpretation has a counterpart in the field philosophy presented by Karl Marx, "... the essence of man is no abstraction inherent in each single individual. In reality, it is the ensemble of the social relations."[6]

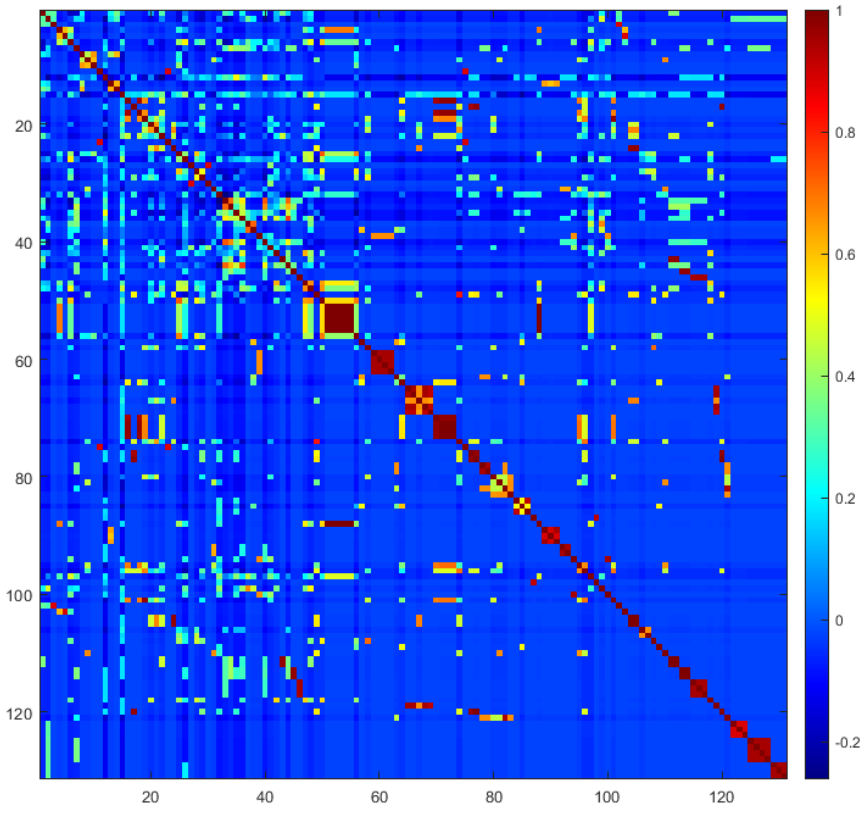


Figure 5: The correlation matrix of the symptoms

Therefore, with the help of MATLAB, we can calculate and plot the correlation matrix for all the symptoms. The x and y axes are the ordinal number of each symptom. Obviously, the diagonal elements are 1 and the

matrix is naturally symmetric. The darker the mesh grid is, the higher correlation the two symptoms will have based on our data set (Fig. 5). Based on our previous derivation, we can do matrix manipulations and obtain the adjusted training set. Notice that all elements in the new training set is continuous (ranging in $0 \sim 1$) rather discrete (0 and 1), which is believed to perform better for supervised learning algorithms like decision trees and neural network.

Finally, we train the decision tree with our new training set with the same decision tree model as previous one. When the maximum splitting number is set as 30, the accuracy of the modified data set is 75.6% while for the previous one, the accuracy is only 69.8%. The visualization of the decision tree models trained by the above two data sets are shown as follows (Fig. 6 and Fig. 7). Apparently, the modified data set is more suitable for pruning of decision tree and should have a higher accuracy as expected.

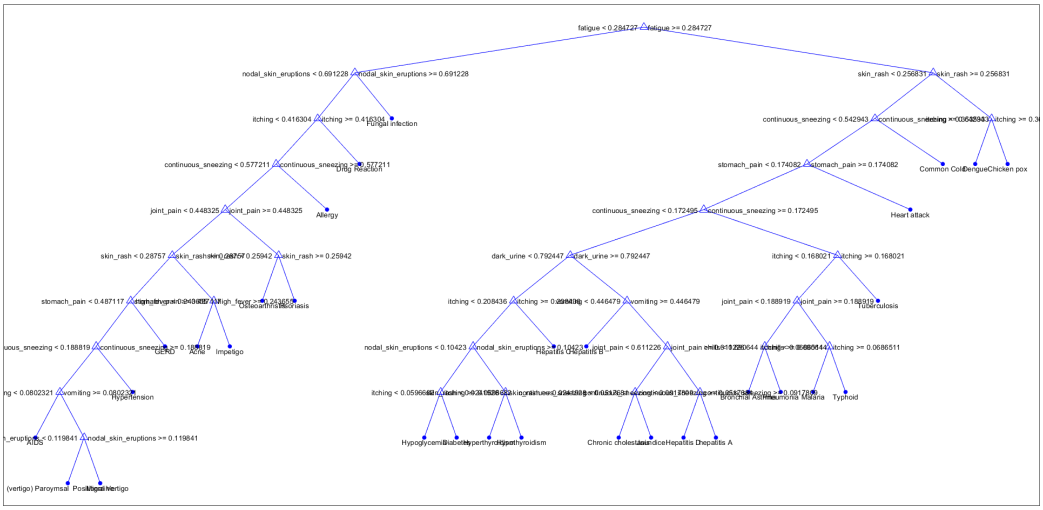


Figure 6: The decision tree plot trained by modified data set.

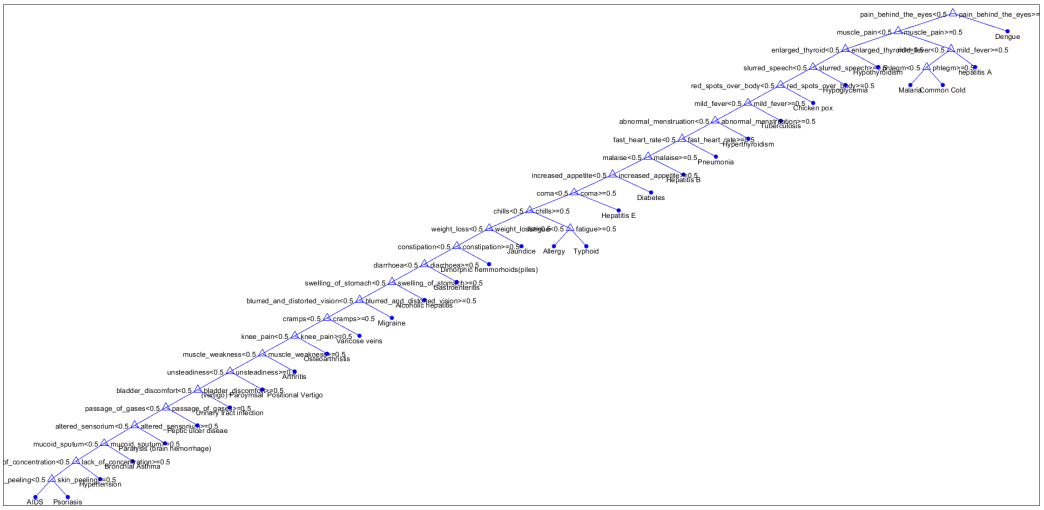


Figure 7: The decision tree plot trained by raw data set.

6.3 Principle Component Analysis (PCA)

Then, since the dimension of our data set is extremely high, we have also tried the Principle Component Analysis (PCA) method to decrease the dimension of raw data set. The PCA algorithm extracts 36 components from the total 132 ones, eliminating 75% of all variables.To name just a few, the top 10 components can explain 12.4%, 10.5%, 6.8%, 6.5%, 5.2%, 3.9%, 3.7%, 3.4%, 3.3% and 2.9% of the variance. In total, these ten components can explain 58.6% of the variance. By doing this, we have successfully improved the accuracy from 69.5% to 75.6% while the maximum splitting number of the decision tree is still set as 30. The PCA can significantly reduce the dimensionality of the data set and therefore save the training time.

6.4 Post-processing with Cross Validation

Usually, we are taught to apply the cross validation to prevent the model from being over fitted. However, is the cross validation always good for the performance of the model? To research this question, we have conducted the experiments on changing the fold N from 0 to 50 and record the corresponding accuracy of the models (decision tree with maximum splitting number of 20). The result that we obtained is listed as follows in Fig. 8. Surprisingly, with the number of fold increasing, the accuracy is decreasing. This may because the size of our data set is too small for the cross validation method to take effect. Instead, since the cross validation is the trade off between availability and accuracy, the performance of the decision tree model becomes worth than before. Therefore, we should think twice before applying the cross validation to the trained model to ensure that it really works.

7 Challenges and Solutions

Our main challenge is how to solve the problem of over fitting because we only have limited-sized raw data and the cross validation method, as we have analyzed before, does not reliable.

Some possible solutions include:

1. Apply the new methods of cross validation that are suitable for small data sets with high dimensionality. For example, we can have a try at the Leave-One-Out Cross Validation (LOO-CV) which is reliable for data sets that are not too large. [5] [7].

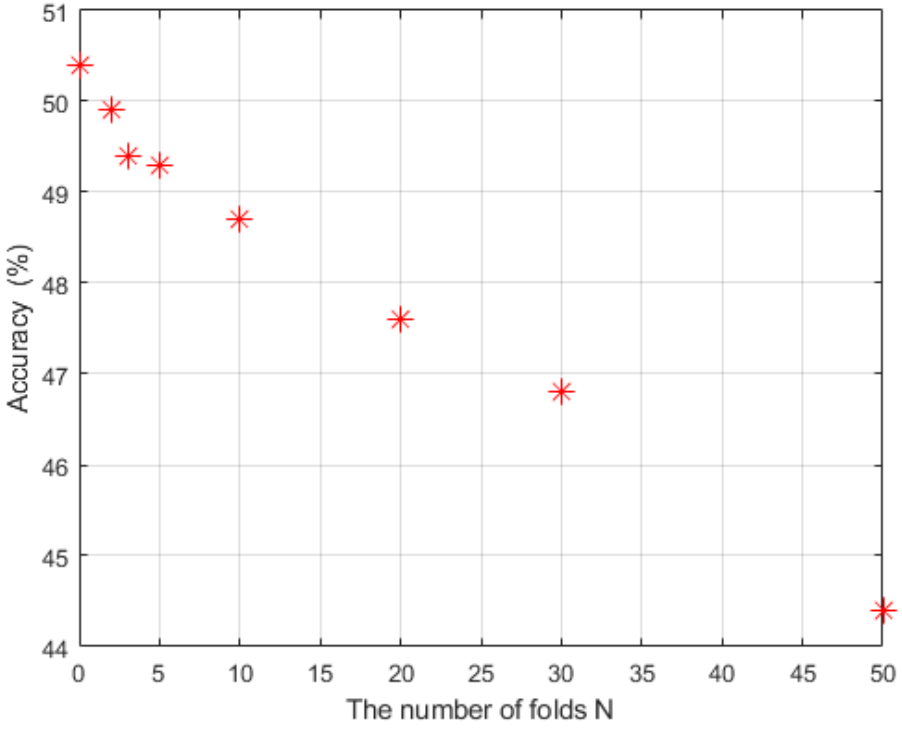


Figure 8: The decision tree plot trained by raw data set.

2. Further apply the boosting tree methods (or extend it into the boosting random forest) to increase both the efficiency and accuracy of decision tree method without being over-fitted. Sometimes sequential fitting of residuals or random selection of certain feature sets may help reduce the probability of over-fitting. [4].

8 Remaining Work

1. Refer to the literature, try to improve the present classification methods such as Ada-boost or random forest of decision tree.

2. Research more about the correlation matrix so as to figure out whether we can intemperate it in a more understandable way.

Appendix

The Matlab Code of Data Processing

```
clc
close all

iter_num=2;
training_mat_cell=cell(1,iter_num);

training=Training(:,1:132);
training_response=Training(:,133);
training_mat=table2array(training);

training_filter=(sum(training_mat)==0);

% training_mat_cell{1}=training_mat(:,¬training_filter);
% for i=1:iter_num-1
% [rho,pval]=corr(training_mat_cell{i});
% %rho(isnan(rho))=0.0000001;
% temp=rho*training_mat_cell{i}';
% temp=(temp-min(temp))./(max(temp)-min(temp));
% training_mat_cell{i+1}=temp';
% end

[rho,pval]=corr(training_mat);
[prow, pcol]=find((pval≥0.3&pval<1));
training_filter(unique(prow))=1;
```

```
training_mat=training_mat(:,¬training_filter);
new_training=array2table(training_mat);
predictorNames = {'itching', 'skin_rash', ...
    'nodal_skin_eruptions', 'continuous_sneezing',...
    'shivering', 'chills', 'joint_pain', 'stomach_pain', ...
        'acidity', 'ulcers_on_tongue',...
    'muscle_wasting', 'vomiting', 'burning_micturition', ...
        'spotting_urination', 'fatigue',...
    'weight_gain', 'anxiety', 'cold_hands_and_feets', ...
        'mood_swings', 'weight_loss',...
    'restlessness', 'lethargy', 'patches_in_throat', ...
        'irregular_sugar_level', 'cough',...
    'high_fever', 'sunken_eyes', 'breathlessness', ...
        'sweating', 'dehydration',...
    'indigestion', 'headache', 'yellowish_skin', ...
        'dark_urine', 'nausea', 'loss_of_appetite',...
    'pain_behind_the_eyes', 'back_pain', 'constipation', ...
        'abdominal_pain', 'diarrhoea',...
    'mild_fever', 'yellow_urine', 'yellowing_of_eyes', ...
        'acute_liver_failure',...
    'fluid_overload', 'swelling_of_stomach', ...
        'swelled_lymph_nodes', 'malaise', ...
    'blurred_and_distorted_vision', 'phlegm', ...
        'throat_irritation', 'redness_of_eyes',...
    'sinus_pressure', 'runny_nose', 'congestion', ...
        'chest_pain', 'weakness_in_limbs', ...
    'fast_heart_rate', 'pain_during_bowel_movements', ...
        'pain_in_anal_region', 'bloody_stool',...
    'irritation_in_anus', 'neck_pain', 'dizziness', ...
        'cramps', 'bruising', 'obesity',...
```

```
41      'swollen_legs', 'swollen_blood_vessels', ...
           'puffy_face_and_eyes', 'enlarged_thyroid',...
42      'brittle_nails', 'swollen_extremeties', ...
           'excessive_hunger', 'extra_marital_contacts',...
43      'drying_and_tingling_lips', 'slurred_speech', ...
           'knee_pain', 'hip_joint_pain', ...
44      'muscle_weakness', 'stiff_neck', 'swelling_joints', ...
           'movement_stiffness', ...
45      'spinning_movements', 'loss_of_balance', ...
           'unsteadiness', 'weakness_of_one_body_side',...
46      'loss_of_smell', 'bladder_discomfort', ...
           'foul_smell_ofurine', 'continuous_feel_of_urine',...
47      'passage_of_gases', 'internal_itching', ...
           'toxic_look_typhos', 'depression', 'irritability',...
48      'muscle_pain', 'altered_sensorium', ...
           'red_spots_over_body', 'belly_pain', ...
           'abnormal_menstruation',...
49      'dischromic_patches', 'watering_from_eyes', ...
           'increased_appetite', 'polyuria',...
50      'family_history', 'mucoid_sputum', 'rusty_sputum', ...
           'lack_of_concentration',...
51      'visual_disturbances', 'receiving_blood_transfusion', ...
           'receiving_unsterile_injections',...
52      'coma', 'stomach_bleeding', 'distention_of_abdomen', ...
           'history_of_alcohol_consumption', ...
53      'fluid_overload1', 'blood_in_sputum', ...
           'prominent_veins_on_calf', 'palpitations',...
54      'painful_walking', 'pus_filled_pimples', 'blackheads', ...
           'scurring', 'skin_peeling',...
55      'silver_like_dusting', 'small_dents_in_nails', ...
```

```
56      'inflammatory_nails', 'blister', ...
            'red_sore_around_nose', 'yellow_crust_ooze'};
57  new_training.Properties.VariableNames=...
58  predictorNames(¬training_filter);
59  new_training=[new_training,training_response];
60
61  imagesc(rho)
62  colormap('jet');
63  colorbar
64  figure
65  imagesc(pval)
66  colormap('jet');
67  colorbar
```

References

[1] Jonathan Bodine and Dorit S. Hochbaum. The max-cut decision tree: Improving on the accuracy and running time of decision trees. In Ana L. N. Fred and Joaquim Filipe, editors, *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2020, Volume 1: KDIR, Budapest, Hungary, November 2-4, 2020*, pages 59–70. SCITEPRESS, 2020.

[2] Syed Fawad Hussain. A novel robust kernel for classifying high-dimensional data using support vector machines. *Expert Syst. Appl.*, 131:116–131, 2019.

[3] Gareth Michael James, Daniela Witten, Trevor J Hastie, and Robert Tibshirani. *An introduction to statistical learning : with applications in R*. Springer, 2013.

[4] Naresh Kumar. Advantages and disadvantages of random forest algorithm in machine learning, 2019.

[5] Sunil Kumar and Ilyoung Chong. Correlation analysis to identify the effective data in machine learning: Prediction of depressive disorder and emotion states. *International Journal of Environmental Research and Public Health*, 15:2907, 12 2018.

[6] Karl Marx. Theses on feuerbach, 2020.

[7] William T. Stephenson and Tamara Broderick. Approximate cross-validation in high dimensions with guarantees. In Silvia Chiappa and

Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 2424–2434. PMLR, 2020.

[8] Hui Zheng, Jing He, Yanchun Zhang, Guangyan Huang, Zhenjiang Zhang, and Qing Liu. A general model for fuzzy decision tree and fuzzy random forest. *Comput. Intell.*, 35(2):310–335, 2019.