Chapter 13
Robust Predictive Models in Clinical Data—Random Forest and Support Vector Machines

Siqi Liu, Hao Du, and Mengling Feng

Abstract In this chapter, we aim to explain the principles that make random forest (RF) and support vector machines (SVMs) successful modelling and prediction tools for a variety of applications. We try to achieve this by presenting the basic ideas of RF and SVMs, together with an illustrative example using the MIMIC III database. The advantages and limitations of both methods are discussed in the chapter. The chapter provides some guidance for choosing a machine learning model, building and training the model, validating model performance and interpreting the results with the Python programming language.

Keywords Predictive model · Mortality prediction · Random forest · Support vector machine

Learning Objectives

- To understand the basic ideas of random forest and support vector machine
- To understand the advantages and limitations while choosing a machine learning model
- To build and evaluate a machine learning model on ICU mortality prediction problem
- To interpret the model results in clinical problems

13.1 Background

In this chapter, we are going to introduce two commonly used statistical models for healthcare data: random forest and support vector machines (SVM). These two models are mainly used for two purposes: first, to create robust and accurate predictive models and second, these models are used in order to evaluate and interpret the

S. Liu · H. Du · M. Feng (✉)
Saw Swee Hock School of Public Health, National University of Singapore and National University Health System,
10-01 Science Drive 2, Singapore, Singapore
e-mail: ephfm@nus.edu.sg

L. A. Celi et al. (eds.), *Leveraging Data Science for Global Health*,
https://doi.org/10.1007/978-3-030-47994-7_13

features (clinical variables). To their advantage, these methods both prevent overfitting and obtain reliable results. Additionally, random forest reduces bias by utilizing average ensemble and SVM uses kerneling to introduce non-linearity. More details will be explained in the following chapter.

In the following chapter and exercises, we are going to explain and demonstrate how these two models work and how to use them.

13.2 Random Forest

Random forest is an ensemble model which fits multiple decision tree classifiers on subsets of the training data and uses averaging to improve the predictive score and control over-fitting (Liaw and Wiener 2002). To understand a random forest, we must start with the basic building block of the model: the decision tree model.

13.2.1 Decision Tree

A decision tree model is a supervised model that learns to predict the expected output by answering a series of questions and making decisions based on the answers (Safavian and Landgrebe 1991). The concept of a decision tree model is subconsciously used by us throughout our daily life. For example, clinicians may intuitively evaluate a patient's condition by asking a series of questions, progressively reaching a diagnostic conclusion. We will use a clinical example to illustrate this concept further: predicting a patient's ICU mortality from first day ICU admission data.

In order to predict ICU mortality, we need to work through a series of queries. We may begin with a reasonable initial question given the domain knowledge, such as asking how old the patient is. In general, the survival rate of a young patient will be higher than the elderly in ICU. After this question, we will look at other predictive variables that could be helpful in determining a patient's ICU mortality, such as the gender of the patient, laboratory results (particularly abnormal values), and treatments the patient may be receiving. By asking these questions, ICU clinicians may garner the likelihood that the patient might survive their ICU stay. A decision tree model would similarly follow that clinical thinking process. It designs a set of questions that segregates the data with each subsequent question, narrowing our possible values until we are confident enough to make a single prediction. This example clinical decision tree is illustrated in Diagram 13.1. The complexity of a decision tree can be represented by tree depth, the number of steps from the root node to the leaf node. In Diagram 13.1, the depth of the decision tree is three. In practical analysis, if the depth is too large, then your model is too complex and you might face an overfitting problem. If the depth is too small, then your model might not capture the variance in data and thus might underfit the problem.

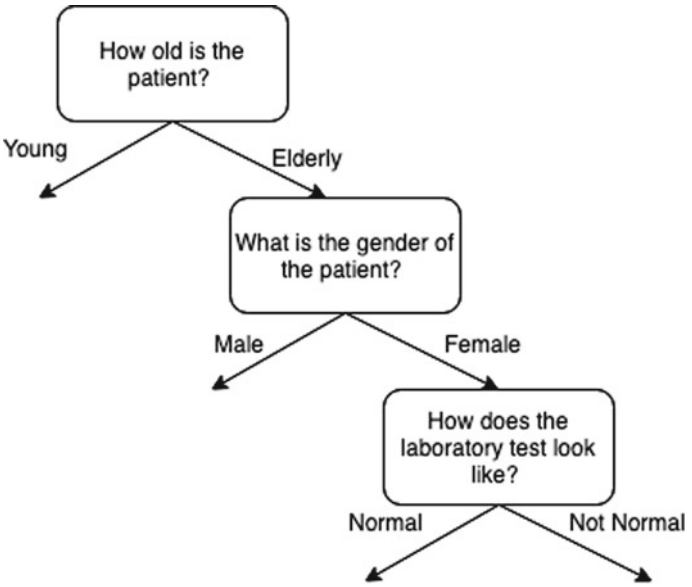

Diagram 13.1 Clinical decision tree

In brief, that is the high-level concept of a decision tree: a flowchart of questions leading to a prediction. Now, we take the mighty leap from a single decision tree to a random forest.

13.2.2 From Decision Tree to Random Forest

Health care is incredibly complex and there are many factors to take into account when clinicians try to make a prediction. Furthermore, every clinician approaches a problem with different background knowledge. Even if they are encountering the same patient, decisions and treatments from any two clinicians may differ from each other. This challenge is similar for decision tree models: if looking at different subsamples of training data, decision models may fit them with different flowcharts of questions and get different conclusions. In technical terms, there exists variance in predictions because they are widely spread around the correct answer. If we collect a group of hundreds or thousands of clinicians, some making the correct prediction and some of making incorrect predictions, we might assume the majority represents best practice and thus take the most popular prediction as the final decision. This is the concept of a random forest. The fundamental idea is to ensemble many decision trees into a single model to reduce the prediction variance. Individually, the prediction from a single human or decision tree may not be accurate, but by combining multiple

observations, the variance is reduced and the predictions are more likely aggregated around the true outcome.

In a random forest model, each decision tree only accesses a random subset of training data. This increases the diversity of the ensemble model, thus improving the robustness of the overall model. That is why we call this model "random." When the model makes a prediction, random forest takes outputs from all individual decision tree models and outputs the prediction with the highest votes among the individual models. In our example here, the ICU mortality prediction is a classification task, where we are predicting a binary outcome of mortality (Death/Survival). In other cases where the targets are a continuous value (such as "ICU Free Days"), we would use a regression analysis and would take the average for the predicted values.

Since we are looking at a binary supervised classification problem in our example (ICU mortality), we may also consider another popular statistical method to model the data: a support vector machine (SVM).

13.3 Support Vector Machines (SVM)

A support vector machine (SVM) is a supervised machine learning algorithm that is used for both classification and regression purposes (Hearst et al. 1998). That said, SVMs are more commonly employed for classification problems, so we will be focusing on SVM with classification problems here.

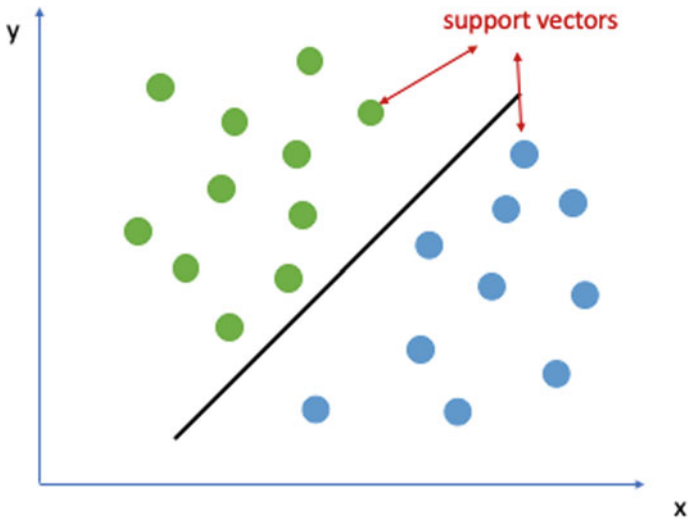

Fig. 13.1 SVM and support vectors

The concept of an SVM is finding a dividing plane that maximizes the margin between two classes in a dataset and achieves the best fit, as illustrated in Fig. 13.1. This plane is called a hyperplane.

Support vectors are the points nearest to the hyperplane. If these points are removed, the position of hyperplane would likely be altered to divide the dataset better. In other words, support vectors are the data points (vectors) that define the hyperplane. Therefore, they are considered to be the critical elements of the dataset.

13.3.1 What is a Hyperplane?

As shown in Fig. 13.1, there are two features for the classification task. The data are in a two-dimensional space, so we can think of a hyperplane as a straight line which classifies the data into two subsets. Intuitively, the farther the data points (support vectors) lie from the hyperplane, the more confident we are that they have been correctly classified. Therefore, the model will place the data points as far away as possible from the hyperplane while making sure the data points are correctly classified. When we feed new data to the SVM model, whatever side of the hyperplane it falls determines the class that it is assigned.

13.3.2 How Can We Identify the Right Hyperplane?

The hyperplane is determined by the maximum margin, which is the distance between a hyperplane and the nearest data point from either class. The goal of fitting an SVM model is to choose a hyperplane with the highest possible margin between the hyperplane and any training data points, which grants the best chance for new data to be classified correctly. We will now illustrate some scenarios on how an SVM model can fit the data and identify the right hyperplane.

Scenario 1 (Fig. 13.2): Here, we have three hyperplanes (A, B and C). In identifying the best hyperplane to classify blue dots and green dots, "hyperplane "B" has clearly best performed the segregation of the two classes in this scenario.

Scenario 2 (Fig. 13.3): If all three hyperplanes (A, B, C) has segregated the classes well, then the best hyperplane will be the one that maximizes the linear distances (margins) between nearest data point for either of the classes. In this scenario, the margin for hyperplane B is higher when compared to both A and C. Hence, we determine the best hyperplane as B. The intuition behind this is that a hyperplane with more considerable margins is more robust; if we select a hyperplane having low margin then there is a higher chance of misclassification.

Scenario 3 (Fig. 13.4): In this scenario, we cannot create a linear hyperplane between the two classes. This is where it can get tricky. Data is rarely ever as clean as our simple examples above. A dataset will often look more like the mixture of dots below - representing a non-linearly separable dataset.

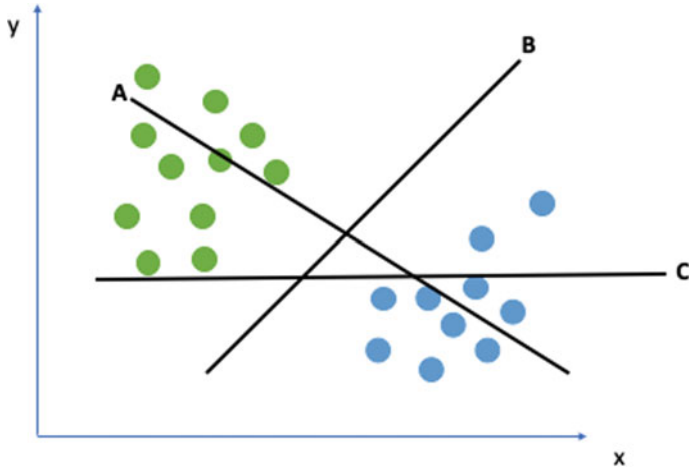

Fig. 13.2 Hyperplane identification, scenario 1

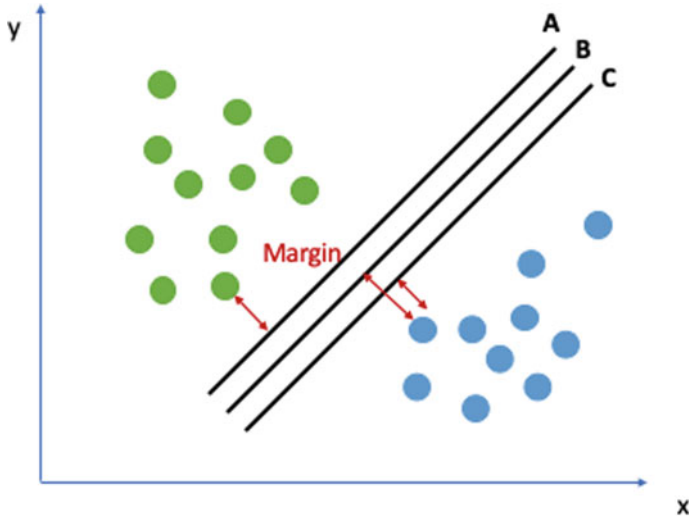

Fig. 13.3 Hyperplane identification, scenario 2

So how can a SVM classify these two classes? With datasets like this, it is sometimes necessary to move away from a 2-dimensional (2-D) view of the data to a 3-D view. Imagine that our two sets of coloured dots above are sitting on a sheet and this sheet is lifted suddenly, launching the dots into the air. While the dots are up in the air, you use the sheet to separate them. This 'lifting' of the dots represents the mapping of data into a higher dimension. This is known as kernelling (Fig. 13.5).

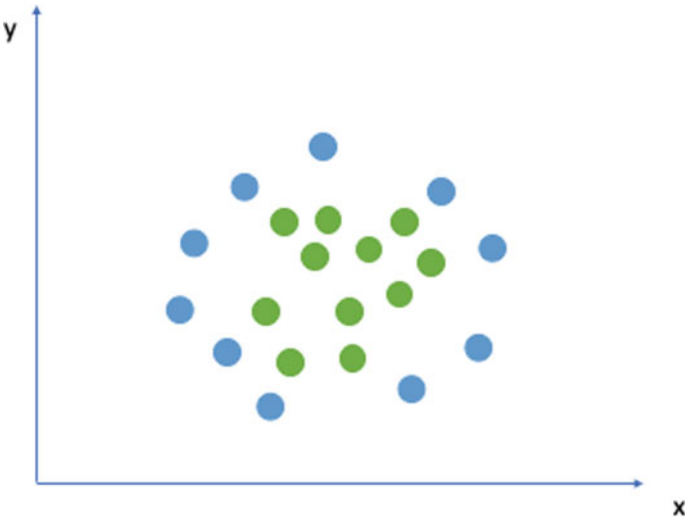

Fig. 13.4 Hyperplane identification, scenario 3

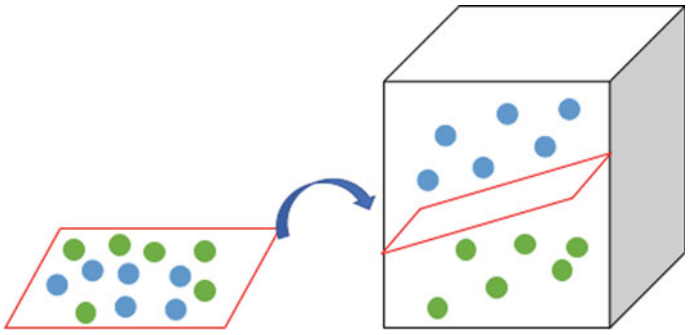

Fig. 13.5 Hyperplane identification, kernelling

Kernelling uses functions to take low dimensional input space and transform it into a higher dimensional space. For example, it converts a non-separable problem into a separable problem for classification. In Fig. 13.4, the kernels map the 2-D data into 3-D space. In the 2-D space, it is impossible to separate blue and green dots with linear functions. After the kernel transformation maps them to 3-D space, the blue and green dots can be separated using a hyperplane. The most commonly used kernels are "linear" kernel, "radial basis function" (RBF) kernel, "polynomial" kernel and others (Hsu et al. 2003). Among them, RBF kernel is the most useful in non-linear separation problems.

In real life scenarios, the dimension of the data can be far higher than 3-D. For instance, we want to use a patient's demographics and lab values to predict the ICU mortality, the number of available predictive variables is easily as high as 10+

dimensions. In this case, the kernel we used in the SVM will require us to map the 10-dimensional data into an even higher dimension in order to identify an optimal hyperplane.

We will demonstrate how to analyse this problem with the above statistical methods using Jupyter Notebook and the Python programming language in the following exercises.

13.4 Limitations of Random Forest and SVM

There are limitations in utilizing random forest for healthcare data. First, these models are sometimes difficult to interpret. For random forest, we may be able to interpret individual decision trees; however interpreting the ensembled random forest model is difficult for complex healthcare data. For SVMs the results are also difficult to interpret due to kernel transformations. Secondly, in tuning the model, some hyper-parameters of these methods need to be determined by users and cannot be optimized.

13.5 Exercise Introduction

In this series of exercises, we will begin to apply statistical learning methods such as random forest and support vector machines (SVM) on real world clinical electronic health records (EHR) data. We are going to use data extracted from a publicly available research clinical dataset, the **M**edical **I**nformation **M**art for **I**ntensive **C**are (MIMIC III). The database contains granular, deidentified ICU data generated from over 70 intensive care unit beds with medical, surgical, cardiac, and neurological patients. Data in MIMIC-III includes demographic information, vital signs, medication records, laboratory measurements, observations, fluid balance, procedure codes, diagnostic codes, imaging reports, hospital length of stay, survival data, and so on.

13.5.1 Jupyter Notebook

Jupyter Notebook is a web application that creates an interactive environment for you to code and view the outputs of your results. It is widely used in data cleaning and transformation, statistical modelling, data visualization, machine learning, etc. Instructions for Jupyter Notebook installation can be found at: http://jupyter.org/install.html. We will be using the Python3 kernel of Jupyter Notebook in this series of exercises.

13.5.2 Data

In this example, the problem we will investigate is prediction of ICU mortality using patient demographics and first laboratory tests. We have the demographic information and first laboratory measurements for patients admitted to the ICU. These information reflect the patient state at the moment they were admitted to ICU and we are going to use them to estimate the probability that the patient is going to survive in ICU. In statistical modeling or machine learning, this is a supervised classification problem. It is a supervised problem because, besides features (demographics and first lab measurements), we also have labels (ICU mortality) for model training. The trained model would be able to classify the patient's condition into ICU survival group or ICU non-survival group, thus making it a classification problem.

13.5.3 Workflow

After the clinical question is defined, we will follow a workflow to analyze and answer the question. The workflow is as follows:

1. Problem statement and data specification
2. Identification and extraction of the required data
3. Data cleaning and missing value processing
4. Data formatting and preparation for modelling
5. Model training
6. Model performance evaluation
7. Parameter adjustment and fine-tuning
8. Model interpretation and report

Our clinical question has been defined above, so we will start from step 2 in exercise 1. These exercises can be found online at: https://github.com/criticaldata/globalhealthdatabook.

References

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., & Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their Applications, 13*(4), 18–28.

Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to support vector classification.

Liaw, A., & Wiener, M. (2002). Classification and regression by random forest. *R News, 2*(3), 18–22.

Safavian, S. R., & Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics, 21*(3), 660–674.