Data Science Toolbox Coursework 3

Kexi Huang, Xiaomeng Jia, Haimeng Lu, Keyu Long

University of Bristol

December 20, 2019

Each member contributes 25% to the total project

Contents

1 Introduction

In today's network world, hundreds of millions of network connections are generated between terminals every day. However, not all network connections contain useful information, including a large number of malicious network attacks (non-normal network flow), so it is very important to monitor and predict the network flows all the time. In this group project, we consider the KDD99 (small, 10%) dataset called kdd-cup.data_10_percent.gz from the week 3 workshop. Our task is to provide a non trivial model and determine an information goal, and finally compare our model results with baseline model.

Through observation, it is found that this dataset contains as many as 490,000 network connections, each of which has 41 different characteristics, representing the first 41 columns of the data set. The last column of the data set indicates the category of each network connection, which can be roughly divided into normal network flow and non-normal network flow. Among them, non-normal network flow includes 22 types such as 'smurf' and 'neptune' [1]. It is worth noting that in the non-normal network flow of this dataset, except 'smurf' and 'neptune', the number of other non-normal types is quite small, so we renamed them as 'OTHER'. After that, we decided to classify and predict the four types of network flow: 'normal', 'smurf', 'neptune' and 'OTHER'.

Our group regards the Decision Tree as the baseline model of this project. Lu's method is Naive Bayes classifier, Huang's method is Multilayer Perceptron, Jia's method is Random Forest, and long's method is Griant Boost Machine. Finally, we use the confusion matrix to show the accuracy of our results and compare them.

2 Data Processing

2.1 Numeric features

2.1.1 Feature Selection

After a quick glance at the summary of the whole data set, we can find that some of the features have small variance compared to the mean value, and that some of them have the majority of data with the same value. In both cases, it implies that the feature does not have enough information for describing the data, and it does little to do the classification task. Therefore, we choose to remove these features using "nzv " function in caret library. This function removes all features with small variances, or with the frequency of a value higher than 95%.

2.1.2 Normalization

In order to eliminate the imbalance in weights of features caused by different magnitude, we have normalized all remaining numeric features using the "preProcess" function in the caret library. Note that the values are all non-negative, therefore we choose to normalize the features using "range" method i.e.

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}.$$

2.2 Categorical features

The basic idea is to use one-hot encoding to encode the categorical features, mapping categorical labels to vectors in $k-$dimensional space, where k is the number of all possible labels. However, the problem is that some of the categorical features have more than 60 labels, which means we have to add 60 columns to the data set, bringing too much extra calculation in the model. To cope with this problem, we firstly relabel the labels with few elements (in this project we set the threshold to 4000) to the label "other". As the result, we have 7 labels in "service" feature, and 4 labels in "normal" feature.

Then we use the one-hot encoding. For a non-numeric feature with m labels, we map each of the labels to e_i, where e_i denotes the i_{th} standard orthogonal basis of $\mathbb{R}^m$. Finally, we have got a data set with 33 features, 7 of which are numeric features, and the rest are encoded from the categorical features.

3 Models

3.1 Baseline Model(CART)

Classification and Regression Trees is a supervised learning that uses one decision tree to find the relationship between the response variable (represented in the branches) and explanatory variables (represented in the leaves). It is often called as a classification tree if the response variable takes a discrete set of values. The critical step is to decide a metric to construct the decision tree by choosing the algorithms used to decide the choice of each branch. Gini impurity, Information gain and Variance reduction are three common metrics. It is easy to understand and works well with large data. It is able to handle both numerical and categorical data and do not require much data preprocessing.

This method is the basic way to do a multi-class problem. It is easy to understand and it can show us some features of the data set. Most of our models are based on decision tree. In the aspect we could choose decision tree as a baseline model to compare the improvement of the methods based on trees.

3.2 Naive Bayes Classifier

3.2.1 Introduction

Naive Bayes is one of a basic technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. Naive Bayes is not a single algorithm for training such classifiers, but a family of algorithms with different prior distribution like Gaussian distribution, Bernoulli distribution and Multinomial distribution. All of these classifiers are based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable.

3.2.2 The mathematical model of NB

Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $\mathbf{x} = (x_1, \ldots, x_n)$ representing some n features (independent variables), it assigns to this instance probabilities $p\left(C_k|x_1, \ldots, x_n\right)$ Using Bayes' theorem, the conditional probability can be decomposed as

$$p\left(C_k|\mathbf{x}\right) = \frac{p\left(C_k\right) p\left(\mathbf{x}|C_k\right)}{p(\mathbf{x})}$$

where $p\left(C_k\right)$ is the prior distribution, and $p\left(C_k|\mathbf{x}\right)$ represents the likelihood.

3.2.3 Code implementation

There are many packages in R helping us constructing Naive Bayes Models. We mainly use the function NaiveBayes in the package klaR. It does not take much time to calculate the model, yet we find that the running time of the predicting of the model is very long.

The accuracy is high. However, most of the modern algorithm has a higher accuracy and less running time than NaiveBayes algorithm. The assumption of Naivebayes Classifier that all of the variables are uncorrelated can be one of the explanations for the result. Whats more, we cannot set different prior distributions in this function, the prior distribution can affect the result of a Bayes classifier.

Then we find a package called fastNaiveBayes. This package can construct different models with various prior distributions. The running time is also faster than others both of constructing the model and making predictions. However, these fast methods only have a high accuracy when making a 0-1 classification.

3.3 Multilayer Perceptron

3.3.1 Background

Multi-Layer Perceptron (MLP) is a class of feedforward artificial neural network (ANN). It can be seen as a logistic regression classifier where the input is first transformed using a learnt non-linear transformation. The basic structure of the model is shown in the graph [figure 1], and there are generally more than three layers in the neural network, which are called input layer, hidden layer(s), and output layer.

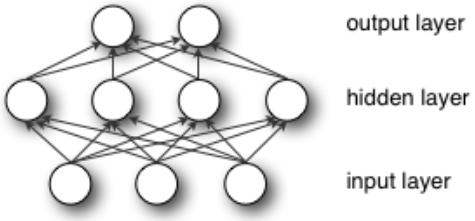

Figure 1: one-hidden-layer MLP

The design inspiration of MLP is derived from the structure of the human brain. The neurons in each layer in the model are analogous to human neurons. When the human brain gets raw information, the neurons of the human brain are activated to extract features and finally calculates the truth of the information. Similarly, when the MLP model obtains raw data as its input, its neurons simulate the processing mechanism of the human brain through a series of linear and non-linear functions, and extract the characteristics of the data and classify them.

3.3.2 Mathematics in Multi-Layer Perceptron

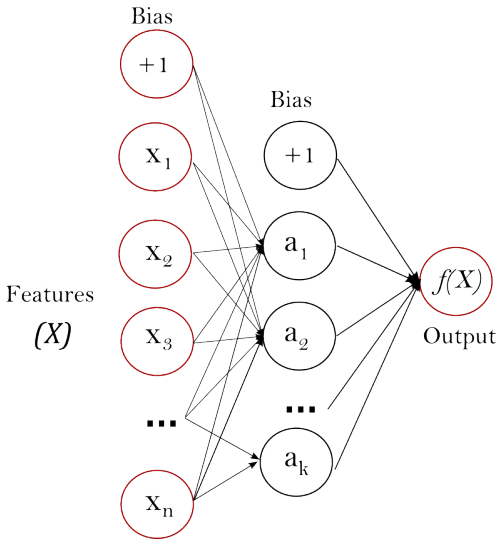

Figure 2: Connection between the layers of MLP

For the mathematical definition of MLP, as shown in the figure above, there are weights w_{ij} between the neurons of two adjacent layers in the MLP, which constitutes a coefficient matrix W of size $k*(n+1)$ connecting these two layers. In the process of spreading from the previous layer to the next, the model calculates a linear combination of neurons of the previous layer and add a bias to it at first

$$a_i' = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + \cdots + w_{in}x_n = W_iX + w_{i0},$$

where w_0 denotes the bias in the previous layer, and W_i denotes the i^{th} row of coefficient matrix W. And then the model applies an activation function to each component (we usually use the Logistic function

as the activation function) to calculate the values of the neurons in the next layer.

$$a_i = f(a_i') = \frac{1}{1 + e^{-a_i'}}.$$

There are such linear and non-linear processes between every adjacent two layers, from the input layer to the output layer.

The model's loss function is Cross-Entropy i.e.

$$Loss(\hat{y}, y, W) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y}) + \alpha \|W\|_2^2,$$

where $\alpha \|W\|_2^2$ is an L_2-regularization term. And in general we use stochastic gradient descent (SGD) to update the parameters, i.e.

$$W^{i+1} = W^i - \epsilon \nabla Loss_W^i,$$

where ϵ is the learning rate with a value larger than 0.

3.3.3 Code implementation

The implementation is done by the existing packet named "RSNNS" and the function "mlp" in the packet. However, the problem is that the output of this model is not a label, but a real number between 0 and 1. As we can see in the prediction, the value for the correct type is very close to 1, and the others are less than 0.1, so we can create a function to decode the output of the model, which is a vector in 4−dimensional space to the 4 labels, referring to "normal" or 3 kinds of attacks. The function assigns 1 to the element with maximum value among 4 values, and 0 to the others, and then decodes it according to the one-hot encoding rule.

The result of the prediction task looks good, with high accuracy up to 99.08%. But the problem is that given the attack type is "other", 40% of the predictions are wrongly calculated as "normal" requests. In practice, it may push the users in risk when wrongly detects an attack as normal requests. The mixture of multiple attacks may cause this bad behavior on the "other" type of attacks, since it mixes the features of the attacks and makes them look like normal request.

3.4 Random Forest

3.4.1 Basic information

As a new and highly flexible machine learning algorithm, random forest (RF) has a wide range of application prospects. Random forest is an algorithm that integrates multiple trees through the idea of ensemble learning. Its basic unit is the decision tree, and its essence belongs to a big branch of machine learning – ensemble learning. Random forest seems easy to understand, in fact, its principle is relatively complicated. To fully understand its working principle, we need a lot of basic knowledge about machine learning. Let's talk about it briefly.

Firstly, we will introduce the sample selection method of random forest, Bagging: Bagging is the abbreviation of Bootstrap AGGregatING. Bagging is based on bootstrap sampling. Given a dataset containing N samples, first take a sample randomly as the training set, then put the sample back to the original dataset, repeat the process N times, and get the training set containing N samples. Some samples in the original data set appear repeatedly in the training set, some never appear. For each tree, the training set is different, and it may contain a large number of repeated samples [3].

Therefore, we can say that Bagging + Decision tree = Random Forest. If we specify the number of trees in the forest, assuming T, we can get T decision tree classifiers. When the random forest is used to predict the prediction set, each tree has its own decision category for each sample, and the final category is generated by simple majority voting.

3.4.2 Key points in random forest

There are two key words in the name of random forest, one is "random", the other is "forest". We have introduced "forest" before, one is called a tree, then hundreds of trees can be called a forest, which is the embodiment of the main idea of random forest – the ensemble learning. "Random" has two meanings. The first one is the randomness of samples, which has been mentioned before. It is generated randomly through bagging. The second is the randomness of features. Suppose that the feature dimension of each sample is M, specify a m much smaller than M, and randomly select m features from M features as a sub feature-set. In this way, the features in the training set corresponding to each tree are also random [3].

3.4.3 Code implementation and model optimization

To implement random forest algorithm, the most important R language package is 'randomForest'. Then we find the randomForest function contains three main parameters: 'mtry', 'ntree' and 'data'. The 'mtry' represents the number of randomly selected features per tree, 'ntree' represents the number of trees in the forest, and 'data' represents the dataset that will run the algorithm. After understanding the meaning of each parameter, we can optimize the model [2].

During the construction of each tree, since bootstrap sampling is carried out for the original dataset, about 1/3 of the samples do not participate in the generation of training set for the k-th tree. We call the remaining 1/3 the OOB (Out Of Bag) samples of the k-th tree. For each sample, about 1/3 of the trees treat it as an OOB sample. These trees are used to predict the category of the OOB sample, and then a simple majority vote is taken as the classification result of the sample. Finally, the ratio of the number of misclassification to the total number of samples in the original dataset is seen as the OOB misclassification rate of the random forest.

First, Jia determines an 'mtry' value to minimize the OOB error rate of the model. He makes a 'for' loop function to generate a random forest in each loop. The loop variable is the number of features selected by each tree, increasing from 1 to M-1(M is the total number of features in the original dataset). Then he look at the OOB error of each forest, find the smallest one(via the 'which.min' function), and use its 'mtry' value(14). (see in Figure 2)

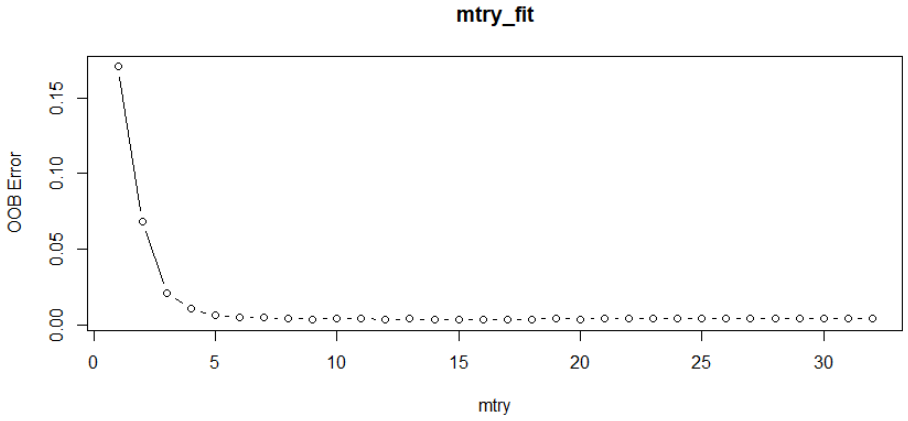

Figure 3: mtry

Next, Jia determine the number of trees in the forest to minimize OOB error. He set a large enough number of trees(1000) as the threshold and plot the graph of the forest. It is found that when 'nTree' is greater than 200, the error of the model tends to be stable. Increasing the number of trees will only increase the computational complexity of the model, but not affect the accuracy. (see in Figure 3)

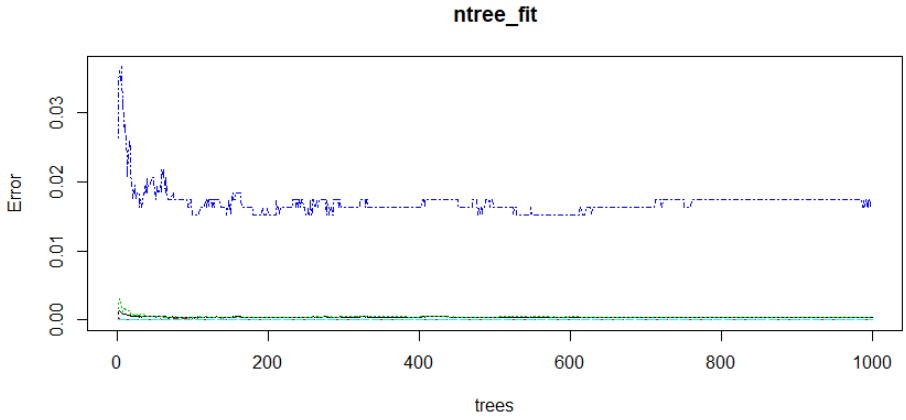

Figure 4: ntree

3.4.4 Analysis of variables' importance

After all the training and prediction work, Jia is interested in the importance of each variable, that is, the impact of each variable on the results. The importance of each variable is measured by their 'MeanDecreaseGini'. The larger the value, the more important the variable is. It can be seen from the graph that the non-numerical variables in the original dataset are very important. Therefore, in order to make the model of other people in the group applicable to the dataset, it is a wise choice to deal with the non-numerical variables with dummy variables [4]. (see in Figure 4)

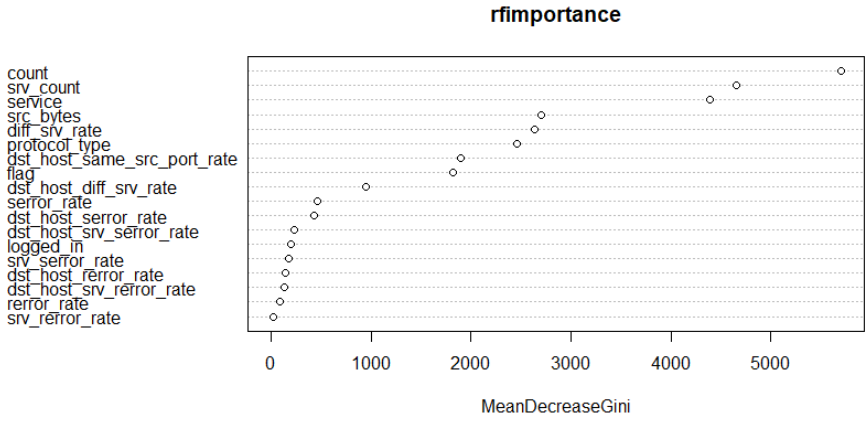

Figure 5: mtry

3.5 Gradient Boost Machine

3.5.1 Background of GBM

Because the effect of a single decision tree model is often not particularly good, people think of integrating multiple decision trees to get the final result. The prediction results under a weak model are limited, so at this time, you can train a second model to learn a mapping of features to residuals (that is, the gap between the output of the model and the real data), and finally we have two models. The results of the predictions are added up to get the final prediction results. But in most cases, the two decision trees cannot get a good result, so we can iterate over the model, which is the origin of the GBM algorithm. [5]

3.5.2 Mathematics in GBM

The GBM algorithm is a method of gradient decent (reducing loss). It uses gradients to improve the model, that is, the value of the negative gradient of the loss function in the current model is used as the approximate value of the residual. That is to say, it is common to operate other loss functions. The purpose of GBM is to reduce each loss through iteration by using the gradient decent method (because the gradient is the fastest growing direction of the function)

$$\text{Loss } = MSE = \sum \left(y_i - y_i^p\right)^2$$

where y_i = ith target value,y_i^p = ith prediction, $L\left(y_i, y_i^p\right)$ is the loss function.
By gradient descent and updating the prediction based on the learning rate, we can find the value with the smallest MSE.

$$y_i^p = y_i^p + \alpha * \delta \sum \left(y_i - y_i^p\right)^2 / \delta y_i^p$$

which becomes $y_i^p = y_i^p - \alpha * 2 * \sum \left(y_i - y_i^p\right)$ where α is learning rate and $\sum \left(y_i - y_i^p\right)$ is sum of residuals. So we update the prediction so that the sum of our residuals is close to 0 (or the minimum value) and the predicted value is close enough to the actual value. [6]

3.5.3 Code implementation

In this experiment, I used the GBM algorithm to predict the types of normal, neptune, smurf, and other (dividing attack types with occurrences less than 4000 into other). Use dummy variables to generate four new features (representing normal, neptune, smurf, and other respectively), and then use the gbm function in the gbm package in R to separately predict the four features, and use the summary function to calculate all the predicted variables Influential factors and the extent of their impact. Here are the conclusions from the GBM package predictions: Count, srcbytes and loggedin influence the normal part most. Srcbytes, service of private and flag of SF influence of neptune part most. Srvcount, service of ecri, srcbytes and count influence of the smurf part most. Srcbytes, service of other and flag of RSTR influence of the other part most.

By using out-of-bag and cross-validation methods to test whether the model is overfit and get the optimal number of ntrees, use the optimal number of ntrees to adjust the model parameters, and compare the methods of oob and cv. The optimal method is selected for prediction, and finally the prediction results are integrated, and the confusion matrix is used to obtain the final prediction results for the four variables. The accuracy obtained using GBM was 0.9972.

4 Comparison and Conclusion

Since we used the same pre-processing in this assessment, this means that we are exploring the same data and analyzing it by using the same data transformation and variable selection.

So our models are comparable. We use the confusion matrix to get the confusion matrix to compare the models. The Confusion matrix is a two-row, two-column table consisting of false positives, false negatives, true positives, and true negatives. It allows us to do more analysis than just being limited to accuracy. Accuracy is not a good measure for the performance analysis of the classifier, because if the data set is imbalanced (the number of data samples of each category is too large), misleading results may occur. [10] However, since our data set seems very balanced, so we can judge our model by accuracy.

The overall accuracy of each model was as follows:

Decision Tree Model (Baseline model) − 0.9741
Naive Bayesian − 0.9791
Multilayer Perception − 0.9907
Random Forest − 0.9995
Gradient Boost Machine − 0.9966

When doing multi-class attack classification problems, random forest has the highest accuracy. In theory, GBM's accuracy should be higher than the random forest (after tuning fit). We think that one of the reasons is that the dummy variable reduces the importance of the variable. In fact, both random forest and GBM are based on the decision tree model. The difference is that random forest uses the bagging method and GBM uses the boost method.

Multi-Layer Perceptron is quite different from the other 3 models in theory. It mimics the neuron structure of the human brain, performs non-linear processing on the data, and finally outputs scores for each classification and data. It is a quantitative method. We believe that the point of high accuracy is the activation functions, which enables MLP to fit complicated functions, and its quantitative property also ensures that it has strong learning and prediction capabilities.

Naive Bayesian's results are slightly higher than the baseline model, so we can conclude that Naive Bayesian's predictions are not particularly good, which can be explained. Because Naive Bayesian's model assumes that the variables are directly irrelevant, but in practice, this is difficult to achieve, especially in network data. In addition, Naive Bayesian took Bernoulli as the prior distribution, but the actual data does not obey Bernoulli's distribution.

In conclusion, since we choose the same variable and the same preprocessing, we can conclude that the model with the highest accuracy performs better in this multi-class prediction problem. However, we cannot ignore that well-performing models (such as GBM) are very expensive, so when making predictions, we must not only consider the accuracy, but also the cost of the model and whether the model used is reasonable.

5 Reference

References

[1] Stolfo, J., Fan, W., Lee, W., Prodromidis, A. and Chan, P.K., 2000. Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection. Results from the JAM Project by Salvatore, pp.1-15.

[2] https://blog.csdn.net/qq_34106574/article/details/82016442

[3] https://blog.csdn.net/daydayup_668819/article/details/81083157

[4] https://www.cnblogs.com/precious-hui/p/10696508.html

[5] https://blog.csdn.net/weixin_42408467/article/details/103513560

[6] https://www.jianshu.com/p/381692dec4e9

[7] Philbin J, Chum O, Isard M, et al. Lost in quantization: Improving particular object retrieval in large scale image databases, Computer Vision and Pattern Recognition, 2008. CVPR 2008, IEEE Conference on, IEEE, 2008: 1-8.

[8] https://scikit-learn.org/stable/modules/neural_networks_supervised.html

[9] http://deeplearning.net/tutorial/mlp.html

[10] https://blog.csdn.net/vesper305/article/details/44927047