STATS 790 - Homework 4

Yicen Li

April 7, 2023

1 Question 1

1.1 Data Description

We explore the Smoke Detection Dataset, which is publicly available on **Kaggle**. The dataset is part of Stefan Blattmann's Real-time Smoke Detection with AI-based Sensor Fusion project, and it comprises various environmental data like air temperature and humidity from several locations. The dataset also provides information on the presence or absence of fire alarms. Consequently, we can employ these environmental factors as predictors and the fire alarm statuses as labels for binary classification tasks, enabling us to assess and compare the performance of different classifiers. Further details about the author's work can be found on his GitHub repository.

1.2 Initial Investigation

Upon examining the raw dataset, we find it is unbalanced, containing 62,630 observations in total. The dataset includes 16 numerical and integer variables, such as air humidity and CO2 concentration, without any categorical or string data types. These variables provide information about the local environment and air quality for the observations. The fire alarm status is indicated by 0 for no alarm and 1 for an active alarm. No missing values are found, and after removing the unnecessary index variable, a heatmap of correlations is created. We can observe some highly correlated pairs like PM1.0 and PM2.5. A random selection of variables is tested for significance on a subset of observations, and outlier detection is conducted based on air temperature. As a result, we remove some non-fire alarm observations with temperatures exceeding two times the interquartile range (IQR). The refined dataset consists of 15 variables and 62,180 observations. In the end, we allocate 70% of the updated dataset as the training set and the remaining 30% as the hold-out set.

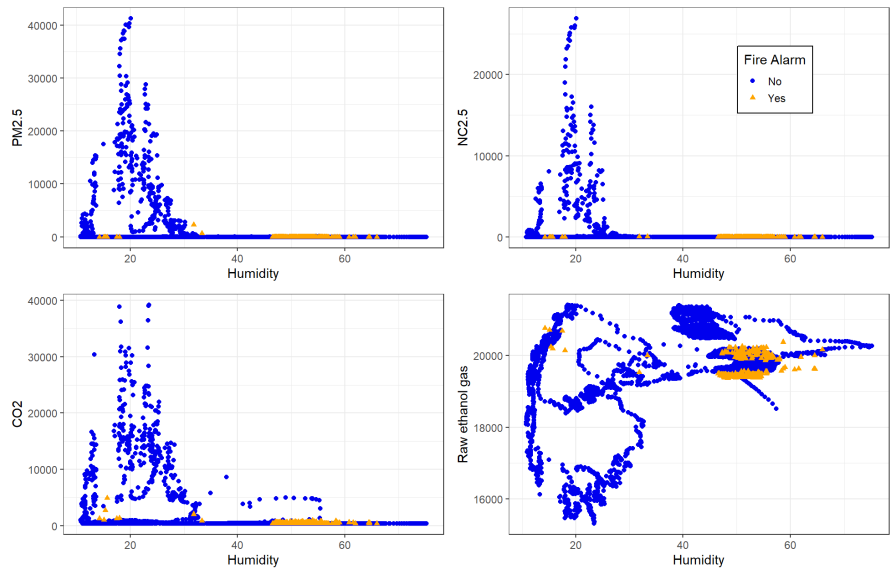

Figure 1: The graphical summaries of the data for selected variables are presented, with the colour and shape indicating the fire alarm status. The x-axis represents air humidity, while the y-axis varies for each subplot. Codes available on Github

1.3 Pre-processing

Feature selection is carried out to improve accuracy and decrease model complexity [3]. Initially, we can remove the CNT variable, which represents the Index. Moreover, by fitting regression models based on the BIC value, the optimal subset is one that excludes PM1.0 and PM2.5. We also attempt to standardize variables to minimize the impact of differing original scales by calculating the root mean square.

1.4 Model Selection

Among tree-based methods such as boosting and bagging, we would like to apply **random forests**. It is because it is a classical machine learning algorithm and is often considered a significant baseline for a classification task. Random Forest offers several advantages over other tree-based methods. Its ensemble approach, which combines multiple decision trees, results in improved accuracy and reduced risk of overfitting. The method is robust, less sensitive to noise and outliers, and can effectively handle missing data without extensive preprocessing. Random Forest also provides a measure of feature importance, enabling the identification of significant variables, and allows for parallelization, which speeds up training on large datasets. Furthermore, the algorithm is stable, resistant to small changes in training data, and versatile, as it can be applied to both regression and classification tasks with equal ease [1].

1.5 Model Tuning

There are several important parameters that need to be adjusted. Firstly, the number of trees in the forest is crucial. We understand that increasing the number of trees can enhance model accuracy but may also raise computation time. We choose 500 as a reasonable balance between accuracy and computational efficiency. Additionally, we establish a (1/3, 2/3) cutoff for determining the winning class since our training set is unbalanced. Furthermore, the maximum depth of the trees can be tuned. Restricting the depth of the trees can help avoid overfitting but may also decrease model accuracy. Through cross-validation, we select 30 as the optimal value.

Random Forest does not have a single, specific loss function. Instead, it combines multiple decision trees as an ensemble method. For classification, decision trees in Random Forest usually use Gini impurity as the splitting criterion in our case.

1.6 Model fitting

According to the above model tuning, we then obtain the best random forest model which achieves 0.0082% misclassification rates on the test set. The following figure suggests the rank of importance provided by our model:

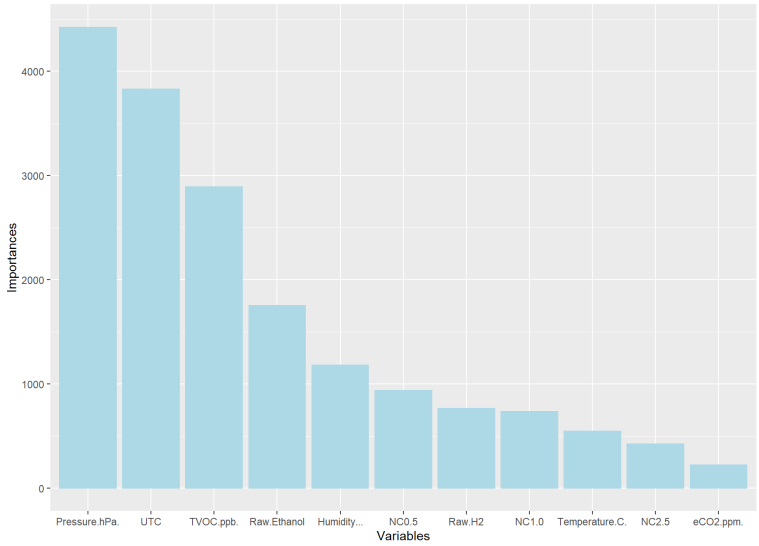

Figure 2: Variable importance is obtained by mean decrease Gini. Codes available on Github

Thus, we can observe that the most significant variable is the air pressure while the least one is the concentration of CO2, according to the random forest model.

2 Question 2

For the gradient boosting algorithm, when the loss function is the mean squared error:

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i,\ f_{m-1}(x_i) + \gamma)$$

Hence we get:

$$\begin{aligned} f(\gamma) &= \sum_{i=1}^{k} L(y_i,\ f_{m-1}(x_i) + \gamma) \\ &= \sum_{i=1}^{k} \frac{1}{k}(y_i - f_{m-1}(x_i) - \gamma)^2 \end{aligned}$$

$$\begin{aligned} \frac{\partial f(\gamma)}{\partial \gamma} &= \frac{2}{k}\sum_{i=1}^{k}(\gamma + f_{m-1}(x_i) - y_i) \\ &= 2\gamma + \frac{2}{k}\sum_{i=1}^{k}(f_{m-1}(x_i) - y_i) \\ &= 0 \end{aligned}$$

Thus we have:

$$\gamma_{jm} = \frac{1}{k}\sum_{i=1}^{k}(y_i - f_{m-1}(x_i))$$

For Newton boosting [2], We have:

$$\begin{aligned} f(\gamma) &= \sum_{i=1}^{k}(L(y_i,\ f_{m-1}(x_i)) + \frac{\partial L}{\partial f_{m-1}}\gamma + \frac{1}{2}\frac{\partial L^2}{\partial f_{m-1}^2}\gamma^2) \\ &= \sum_{i=1}^{k}(\frac{1}{k}(y_i - f_{m-1}(x_i))^2 - \frac{2}{k}(y_i - f_{m-1}(x_i))\gamma + \frac{1}{k}\gamma^2) \end{aligned}$$

Therefore:

$$\begin{aligned} \frac{\partial f(\gamma)}{\partial \gamma} &= -\frac{2}{k}\sum_{i=1}^{k}(y_i - f_{m-1}(x_i)) + 2\gamma \\ &= 0 \end{aligned}$$

and

$$\gamma_{jm} = \frac{1}{k}\sum_{i=1}^{k}(y_i - f_{m-1}(x_i)).$$

For the gradient boosting algorithm, when the loss function is the binomial deviance loss function:

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i,\ f_{m-1}(x_i) + \gamma)$$

We have:

$$\begin{aligned} f(\gamma) &= \sum_{i=1}^{k} L(y_i,\ f_{m-1}(x_i) + \gamma) \\ &= \sum_{i=1}^{k} -log(1 + e^{-2y_i(f_{m-1}(x_i)+\gamma)}) \end{aligned}$$

Then we get:

$$\begin{aligned} \frac{\partial f(\gamma)}{\partial \gamma} &= \sum_{i=1}^{k} \frac{2y_i exp(-2y_i(f_{m-1}(x_i) + \gamma))}{1 + exp(-2y_i(f_{m-1}(x_i) + \gamma))} \\ &= 0 \end{aligned}$$

to solve γ_{jm}. For Newton boosting, We have:

$$\begin{aligned} f(\gamma) &= \sum_{i=1}^{k} (L(y_i,\ f_{m-1}(x_i)) + \frac{\partial L}{\partial f_{m-1}}\gamma + \frac{1}{2}\frac{\partial L^2}{\partial f_{m-1}^2}\gamma^2) \\ &= \sum_{i=1}^{k} (-log(1 + e^{-2y_i(f_{m-1}(x_i))}) + \frac{2y_i exp(-2y_i(f_{m-1}(x_i)))}{1 + exp(-2y_i(f_{m-1}(x_i)))}\gamma \\ &+ \frac{4y_i^2 exp(-2y_i(f_{m-1}(x_i)))(1 + exp(-2y_i(f_{m-1}(x_i)))) - 4y_i^2 exp2(-2y_i(f_{m-1}(x_i)))}{(1 + exp(-2y_i(f_{m-1}(x_i))))^2}\gamma^2) \end{aligned}$$

Then we can obtain γ_{jm} through $\frac{\partial f(\gamma)}{\partial \gamma} = 0$.

References

[1] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.