STATS 780 Project Report: Predicting Mining-Based Seismic Bumps

Tommy Flynn (400121496)

08 December, 2022

Contents

1 Introduction

Mining is a crucial sector of the economy providing many jobs and raw materials. As such, it is important to continuously maintain operations. One particularly troublesome impediment is mining-based seismic bumps (SB) which are akin to earthquakes. They are responsible for loss of life and resources making mining a notoriously dangerous enterprise. In order to minimize loss of life and resources we must accurately predict SBs in advance. Currently, there is no method to reliably predict the occurrence, strength, or duration of general earthquakes in advance [3]. This is because earthquakes are chaotic complex systems occurring deep within the Earth under conditions that are difficult to measure and understand. It will take significant innovation in our understanding of geology before such methods are possible. Our goal for this project is much more modest. We are interested in classifying mining shifts as hazardous or not based on the seismic conditions of the previous shift. The data is sourced from the UCI Machine Learning Repository and contains seismic measurements of a Polish coal mine during 8-hour shifts [4]. The shift was labeled as hazardous if an SB occurred at an energy level of 10^4J or higher and non hazardous otherwise. Classical statistical methods have been applied to this problem but have proven to be insufficient. Thus, we are interested in applying machine learning to accurately predict mining-based SBs. Because the data is labeled into binary classes, we will use supervised learning in the form of binary classification.

2 Methods

As mentioned above, we will be performing supervised machine learning via binary classification. In particular we will apply two algorithms, namely, random forest (RF) and neural network (NN) which are described below.

2.1 Random Forest [2]

RF is an ensemble method made from a collection of decision trees. In our case, we will use classification decision trees as opposed to regression decision trees. Each tree recursively partitions the predictor space with binary splits chosen by optimizing the Gini index. Lower values of the Gini index indicate the split is more pure as it contains more observations from a single class. The ensemble is generated using a special type of bagging or bootstrap aggregating. The bagging procedure fits each classification tree using a bootstrap sample of the training set which is a subset sampled with replacement. The class predictions of each tree are then aggregated. RFs modify bagging by using only a random sample of the predictors per split. Bagging reduces the variance of individual trees and RF decorrelates the trees. We will apply RF without any transformations to the data. The two main hyperparameters in RF are the number of trees and the number of predictors consider per split. To select the optimal values, we will perform grid search with 5-fold cross validation. The grid will contain 100 to 500 trees and 2 to 10 predictors.

2.2 Neural Network [1]

NN is a forward feeding computing system made from connected layers. It is forward feeding because information enters the input layer, travels through a collection of hidden layers, and then exits the output layer. Each layer contains a number of nodes called neurons which are densely connected by edges i.e. each neuron in one layer is connected to all neurons in the next. The edges have an associated weight and each neuron has an associated bias and activation function. The input of a neuron is the product of the previous neurons and their weights plus the bias. This is called the transfer potential. The output of a neuron is its activation function at the value of its transfer potential. This mimics biological neural networks which fire neural impulses given sufficient stimulation. The number of neurons in each layer varies but typically monotonically decreases between the number of predictors and the number of classes. Dropout layers can also be used to eliminate neurons at a specified rate for regularization. Weights and biases are updated by optimizing a loss function using some version of gradient descent. Backpropagation efficiently updates the parameters selected by gradient descent from back-to-front by exploiting the chain rule to avoid duplicate calculations. To apply NN to our data we will one-hot encode the categorical variables, perform statistical standardization, and perform a tensor transformation. The main hyperparameters are the number of hidden layers, the number of neurons per layer, and the dropout rate. The optimal values will be selected via manual search due to the large dimension of the hyperparameter space. The rectified linear unit (ReLU) activation function will be used at all layers except the output layer which will use softmax to assign the probability of belonging to each class. Training will use 20% validation, 30 epochs, 256 batch-size, ADAM gradient descent, and the cross entropy loss function.

2.3 General Procedure

First, the labeled dataset will undergo a stratified 80/20 train-test split. Next, we will use the training set to tune each algorithms' hyperparameters and then fit the optimal models. Subsequently, we will use the test set to evaluate the performance of the optimal models. Model performance will be determined using sensitivity and specificity analysis, accuracy, and F1 score. Sensitivity and specificity analysis selects the optimal cutoff that maximizes the area under the receiver operating characteristic curve, accuracy is the proportion correct predictions, and the F1 score is a penalized accuracy useful for unbalanced classes. F1 score is often defined as the harmonic mean of precision and recall. All measures are described in Table 1 where

TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

Table 1: Performance measures.

Name	**Description**
Sensitivity/Recall	$\frac{\text{TP}}{\text{TP+FN}}$
Specificity	$\frac{\text{TN}}{\text{TN+FP}}$
Precision	$\frac{\text{TP}}{\text{TP+FP}}$
Accuracy	$\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$
F1 Score	$\frac{2}{\text{Precision}^{-1}+\text{Recall}^{-1}}$

The algorithms will be compared primarily on F1-score due to high class imbalance and secondarily on accuracy, sensitivity, and specificity.

3 Results

3.1 Data

The dataset is tabular with 2584 observations and 19 variables recorded during 8-hour mining shifts. Of the 19 variables, 4 are categorical predictors, 14 are numerical predictors, and the remaining is the binary response. The categorical predictors consist of 3 qualitative hazard assessments that rank the shift as low to high hazard based on different criteria and the remaining describes the type of mining work performed. The numerical predictors are summary statistics of the SBs and the most active geophone (GMax) which is a vibration detector. The first 10 pertain to SBs, namely, SB count at 7 different energy levels, maximum energy, total energy, and total count. The remaining 4 pertain to GMax, namely, energy, deviation of energy, pulse count, and deviation of pulse count. The binary response takes a value of 1 if a high energy ($E \geq 10^4$J) SB occurred in the next 8-hour shift and 0 otherwise.

3.2 Exploratory Data Analysis

Recall that there are 2584 observations, 4 categorical predictors, 14 numerical predictors, and a binary response. The 3 categorical variables pertaining to qualitative hazard assessment have four levels which are (a) lack of hazard, (b) low hazard, (c) high hazard, and (d) danger state. There are a decreasing amount of observations in each state. Some levels have no observations which will be relevant for one-hot encoding. There are two levels for the type of mining work which are (W) coal getting and (N) preparation shift. The shifts are mostly coal getting. This can all be seen in Figure 1 which displays various barplots. There are 170 high energy SB occurrences meaning the data is highly unbalanced with about 93% of the observations identified as non-examples. The SB counts at various energy levels take integer values ranging from 0 to 9 with the majority of values being zero. Pulse counts are in the thousands. The largest three SB energy level counts are columns of zeros because no SBs were observed at those levels, so they will be removed leaving us with 11 numerical predictors. The energy values range from 0 to 2595650J with means in the thousands. The deviances range from -96 to 1245. More specific behavior can be seen in Figure 2 below which displays various boxplots. There are not any overwhelming salient differences between classes in the boxplots. Outliers are present because of the massive energy values the occur on occasion which is key information when considering hazards so we will not drop these values. There is moderate positive correlation between the SB variables and GMax variables respectively which makes sense as they are capturing different parts of the same types of processes. Lastly, there were no missing values. To improve the performance of the classifiers, we will use a subset of the data with all 170 examples and 340 randomly sampled non-examples. This improves the class imbalance from 7% to 33%.

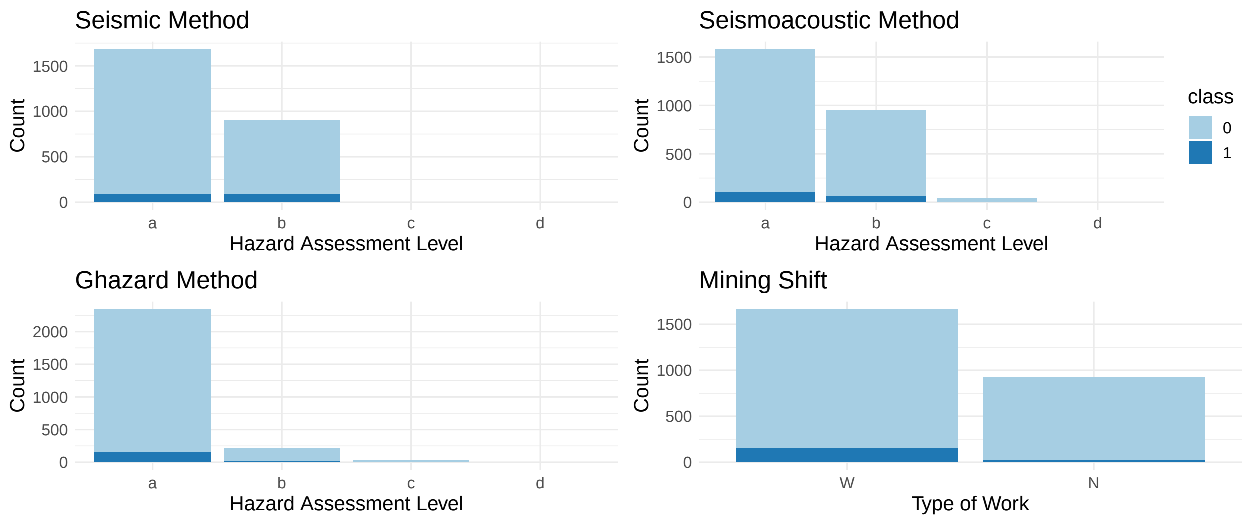


Figure 1: Barplots for categorical features (Hazard = 1, Non-hazard = 0).

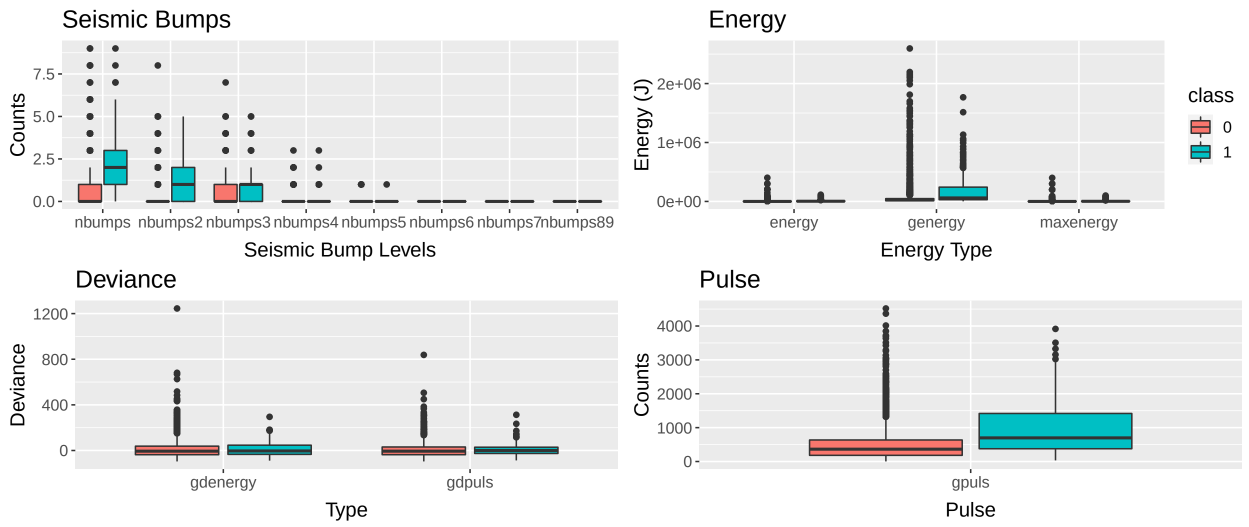


Figure 2: Boxplot pairs for all features (Hazard = 1, Non-hazard = 0).

3.3 Model Performance

The optimal RF selected by grid search with 5-fold cross validation has 400 trees and 2 predictors per split. Sensitivity and specificity analysis selected a cutoff of 0.39 resulting in 0.68 sensitivity and 0.84 specificity. RF accuracy and F1 score are 0.78 and 0.68 respectively. Table 2 shows the RF confusion matrix from which these measures were calculated. Figure 3 shows the RF variable importance plot. With respect to mean decrease in accuracy and Gini index, the variables relating to energy are the most important for classifying hazardous mining shifts.

Table 2: RF confusion matrix (Hazard = 1, Non-hazard = 0).

		Predicted	
		0	1
Actual	0	57	11
	1	11	23

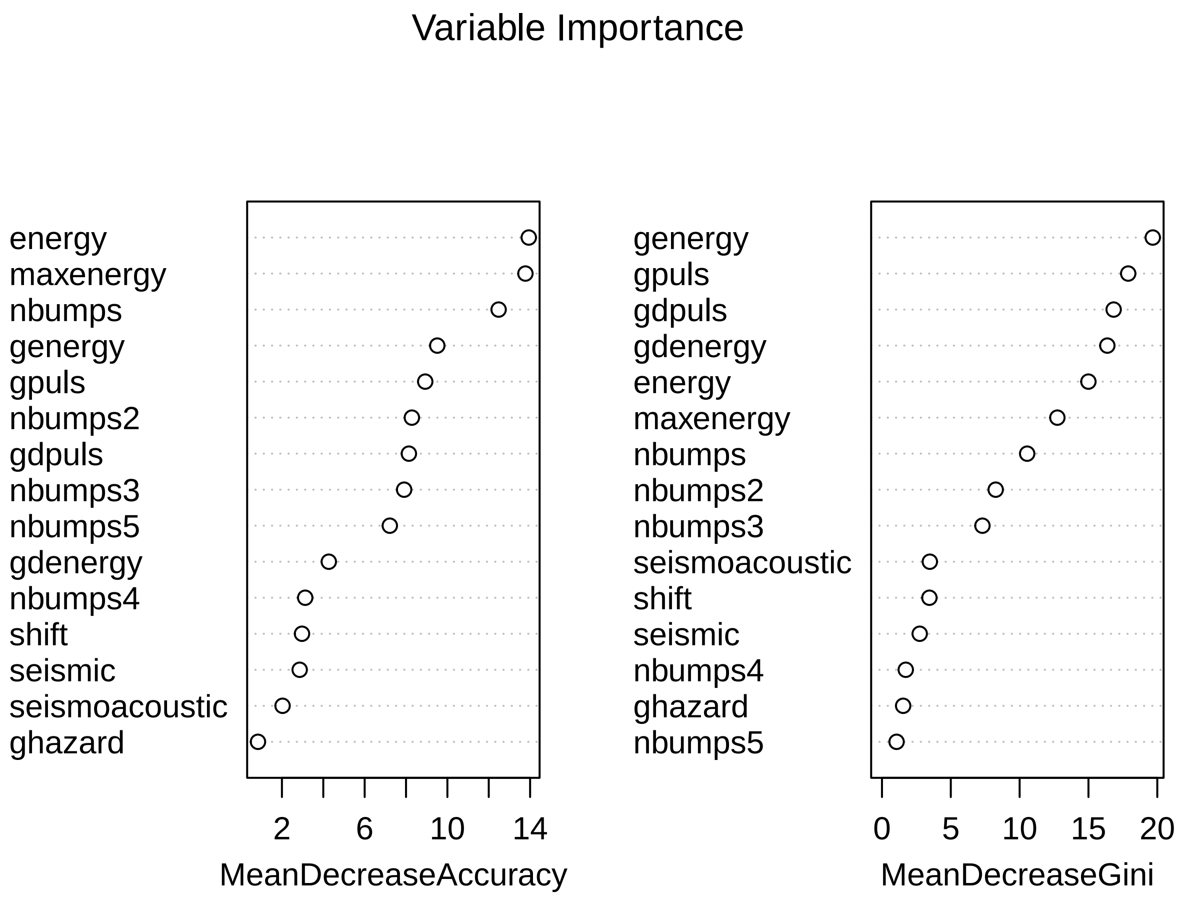


Figure 3: RF variable importance plot.

The optimal NN selected by manual search has a 10 neuron input layer with 0.25 dropout rate, a 4 neuron hidden layer, and a 2 neuron output layer. Sensitivity and specificity analysis selected a cutoff of 0.51 resulting in 0.76 sensitivity and 0.81 specificity. NN accuracy and F1 score are 0.79

and 0.71 respectively. Table 3 shows the NN confusion matrix. Figure 4 shows the training details which terminated at 30 epochs due to the plateau in decrease of loss.

Table 3: NN confusion matrix (Hazard = 1, Non-hazard = 0).

		Predicted	
		0	1
Actual	0	55	13
	1	8	26

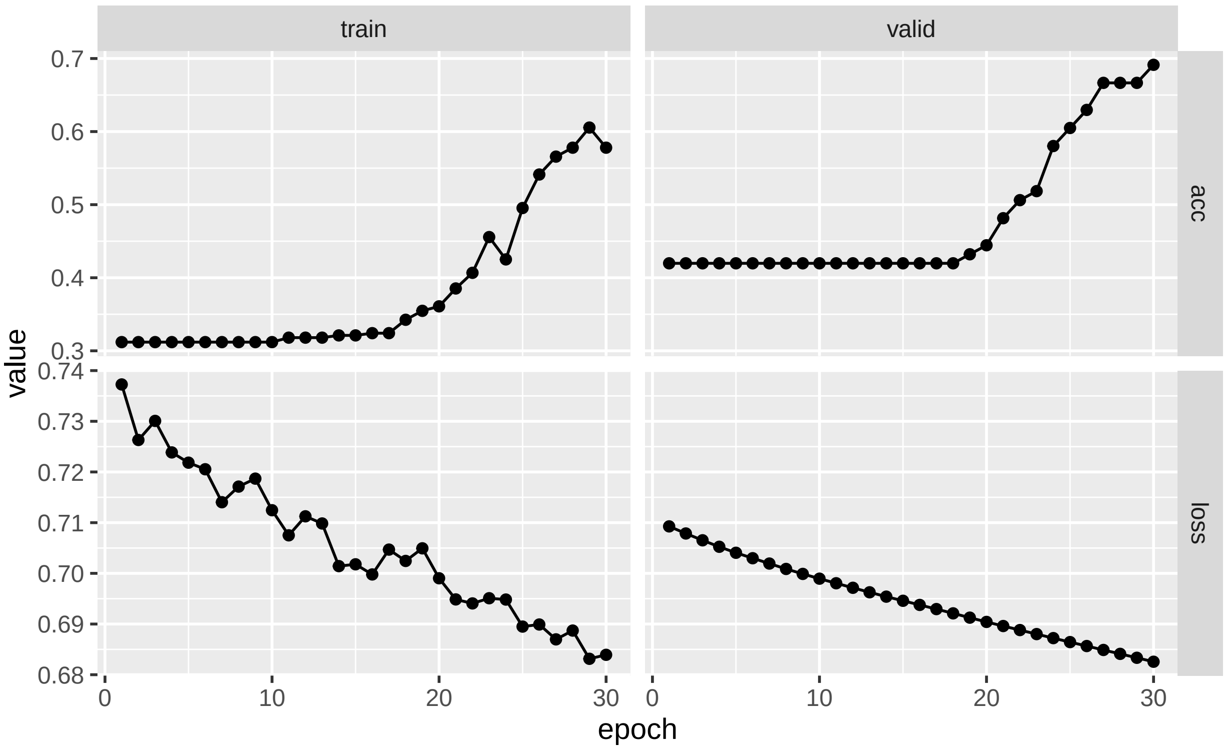


Figure 4: NN training plot.

A comparison of model performance with respect to the aforementioned measures is displayed in Table 4. Notably, NN outperformed RF on the primary and most secondary measures. In particular, NN has 3% superior F1 score, 1% superior accuracy, and 8% superior sensitivity. RF has 3% superior specificity. Thus, NN is the marginally better than RF with superior ability to detect the unbalanced class of high energy SB at the negligible loss of 3% specificity.

Table 4: RF and NN model performance comparison.

Model	**F1**	**Accuracy**	**Sensitivity**	**Specificity**
RF	0.68	0.78	0.68	0.84
NN	0.71	0.79	0.76	0.81

4 Conclusions

Overall, supervised machine learning is a promising tool for seismic hazard classification. Both NN and RF are moderately successful at classifying high energy SB mining shifts with respect to F1 score and accuracy reaching near 0.7 and 0.8 respectively. This is quite impressive considering how difficult seismic prediction is, however, we must recognize that the predictions were made using data pertaining to mere hours before the events. From the RF, we also identify the most useful predictors to be energy related and the least useful to be the qualitative hazard assessments.

In terms of the algorithms, both have pros and cons. The NN is superior in prediction as it achieved higher performance measures of F1-score, accuracy, and sensitivity, however, it is a black box with no interpretability. On the other hand, the RF has interpretability providing insight into the importance of variables. For these reasons, it might be warranted to train both classifiers given sufficient resources.

The two main challenges of this project were the imbalance of the classes and the difficulty of tuning the neural network. Because of the high class imbalance, the algorithms were inclined to learn to classify everything as a non-example to achieve high accuracy. We attempted to solve this problem by using a less imbalanced subset with 33% examples and using the F1-score. Additionally, the neural network was difficult to tune because of the high dimension of the hyperparameter space. We used manual search to achieve a fairly competent model but are unconvinced it is the best model for this dataset. With more computational resources (GPU) it would be interesting to see if a more sophisticated tuning algorithm like Bayesian optimization could achieve superior performance.

References

[1] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT Press.

[2] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: With applications in R.* New York :Springer.

[3] O'Reilly, S., Rawling, T. (2017). *How do you predict an earthquake?* Australian Academy of Science. Retrieved December 2022, from https://www.science.org.au/curious/earth-environment/ea rthquakes

[4] Sikora, M., Wrobel, L. (2013). *seismic-bumps Data Set*, UCI Machine Learning Repository. Retrieved December 2022, from https://archive.ics.uci.edu/ml/datasets/seismic-bumps (2022)

Supplementary Material

```r
# set up
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(ggpubr)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(e1071)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(torch)
library(luz)
library(tictoc)
set.seed(24)
torch_manual_seed(13)



##########################
#  exploratory analysis  #
##########################


# load data
Seismic = read.csv("/Users/tommyflynnrogers.com/Desktop/SeismicBumps.csv")


# first 6 rows
head(Seismic)
```

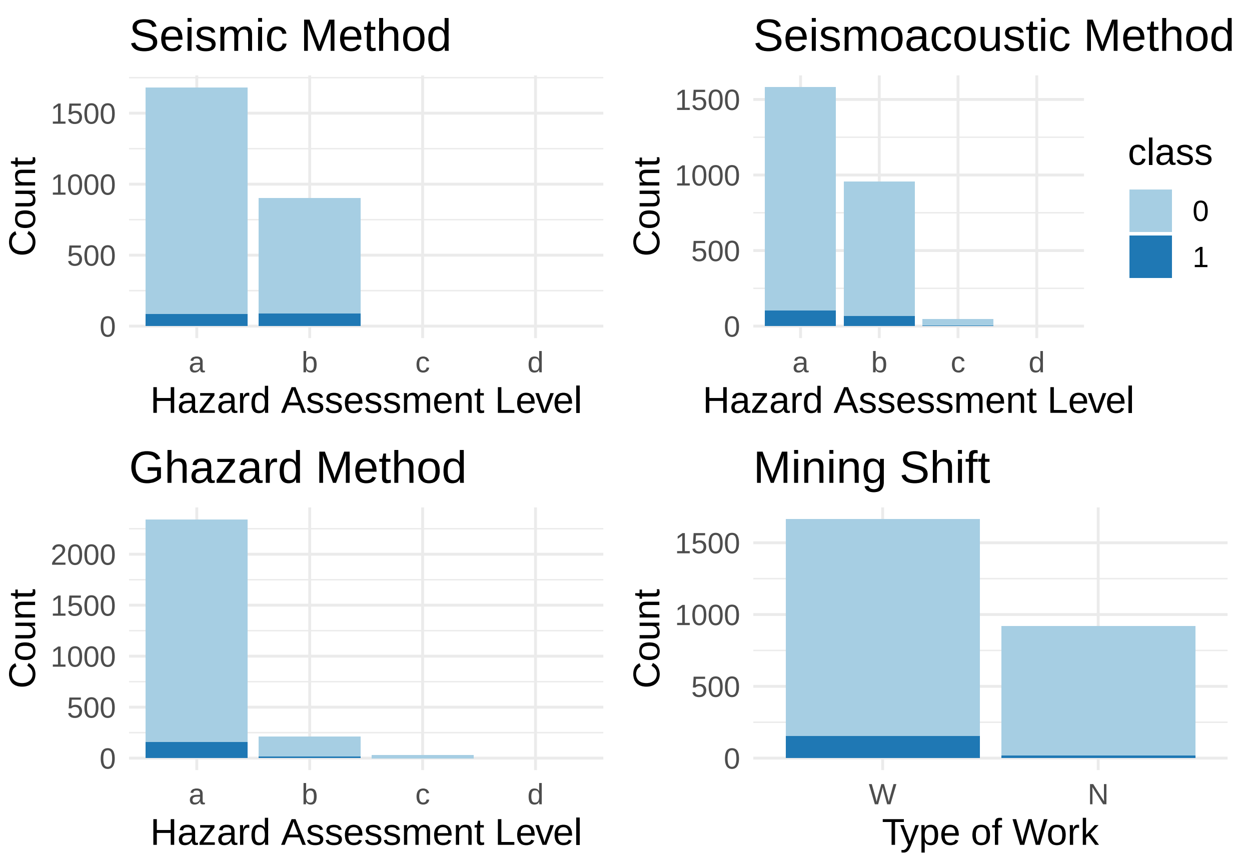
```
##   id seismic seismoacoustic shift genergy gpuls gdenergy gdpuls ghazard nbumps
## 1  1       a              a     N   15180    48      -72    -72       a      0
## 2  2       a              a     N   14720    33      -70    -79       a      1
## 3  3       a              a     N    8050    30      -81    -78       a      0
## 4  4       a              a     N   28820   171      -23     40       a      1
## 5  5       a              a     N   12640    57      -63    -52       a      0
## 6  6       a              a     W   63760   195      -73    -65       a      0
##   nbumps2 nbumps3 nbumps4 nbumps5 nbumps6 nbumps7 nbumps89 energy maxenergy
## 1       0       0       0       0       0       0        0      0         0
## 2       0       1       0       0       0       0        0   2000      2000
## 3       0       0       0       0       0       0        0      0         0
## 4       0       1       0       0       0       0        0   3000      3000
## 5       0       0       0       0       0       0        0      0         0
## 6       0       0       0       0       0       0        0      0         0
##   class
## 1     0
## 2     0
## 3     0
## 4     0
## 5     0
## 6     0
```

```
# drop id column
Seismic = Seismic[,2:20]


# convert to factor
assessment_levels = c('a', 'b', 'c', 'd')
Seismic$class = as.factor(Seismic$class)
Seismic$seismic = factor(Seismic$seismic,
                         levels=assessment_levels)
Seismic$seismoacoustic = factor(Seismic$seismoacoustic,
                                levels=assessment_levels)
Seismic$ghazard = factor(Seismic$ghazard,
```

```r
                         levels=assessment_levels)
Seismic$shift = factor(Seismic$shift, levels=c('W', 'N'))

# summary of data
summary(Seismic)
```

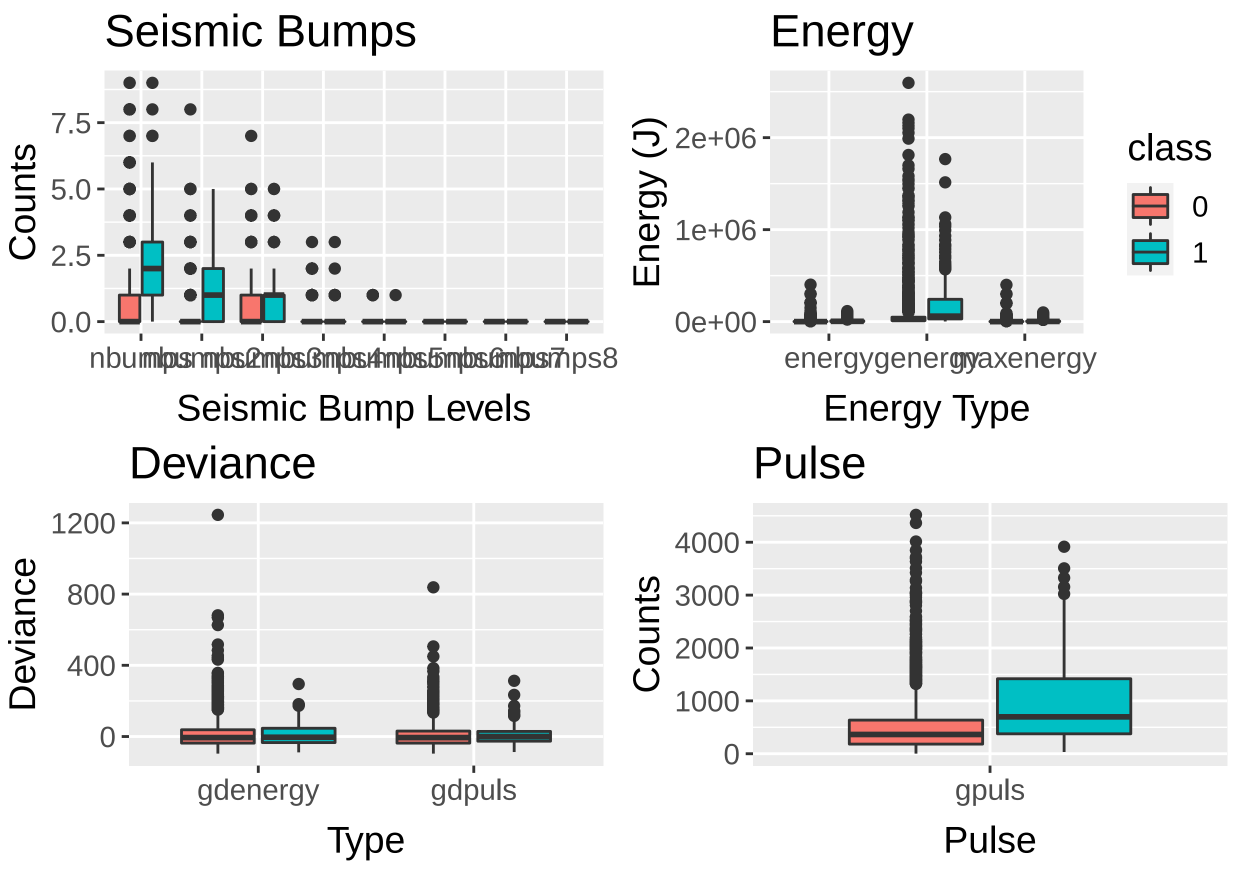
```
##  seismic  seismoacoustic shift       genergy             gpuls       
##  a:1682   a:1580         W:1663   Min.   :    100   Min.   :   2.0  
##  b: 902   b: 956         N: 921   1st Qu.:  11660   1st Qu.: 190.0  
##  c:   0   c:  48                  Median :  25485   Median : 379.0  
##  d:   0   d:   0                  Mean   :  90242   Mean   : 538.6  
##                                   3rd Qu.:  52832   3rd Qu.: 669.0  
##                                   Max.   :2595650   Max.   :4518.0  
##     gdenergy            gdpuls          ghazard      nbumps           nbumps2      
##  Min.   : -96.00   Min.   :-96.000   a:2342   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.: -37.00   1st Qu.:-36.000   b: 212   1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :  -6.00   Median : -6.000   c:  30   Median :0.0000   Median :0.0000  
##  Mean   :  12.38   Mean   :  4.509   d:   0   Mean   :0.8595   Mean   :0.3936  
##  3rd Qu.:  38.00   3rd Qu.: 30.250            3rd Qu.:1.0000   3rd Qu.:1.0000  
##  Max.   :1245.00   Max.   :838.000            Max.   :9.0000   Max.   :8.0000  
##     nbumps3           nbumps4            nbumps5              nbumps6     nbumps7 
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.000000   Min.   :0   Min.   :0  
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0   1st Qu.:0  
##  Median :0.0000   Median :0.00000   Median :0.000000   Median :0   Median :0  
##  Mean   :0.3928   Mean   :0.06772   Mean   :0.004644   Mean   :0   Mean   :0  
##  3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:0   3rd Qu.:0  
##  Max.   :7.0000   Max.   :3.00000   Max.   :1.000000   Max.   :0   Max.   :0  
##     nbumps89     energy           maxenergy       class   
##  Min.   :0   Min.   :     0   Min.   :     0   0:2414  
##  1st Qu.:0   1st Qu.:     0   1st Qu.:     0   1: 170  
##  Median :0   Median :     0   Median :     0           
##  Mean   :0   Mean   :  4975   Mean   :  4279           
##  3rd Qu.:0   3rd Qu.:  2600   3rd Qu.:  2000
##  Max.   :0   Max.   :402000   Max.   :400000
```

```r
#############################    bar plots    #############################
font_size = 14
seismic_barplot = ggplot(data = Seismic) + scale_x_discrete(drop=FALSE) +
  geom_bar(mapping = aes(x = seismic, fill=class)) + theme_minimal() +
  ylab('Count') + xlab('Hazard Assessment Level') +  ggtitle('Seismic Method')+
  scale_fill_brewer(palette="Paired") +
  theme(text = element_text(size = font_size), legend.position="none")

seismoacoustic_barplot = ggplot(data = Seismic) + scale_x_discrete(drop=FALSE) +
  theme_minimal() + geom_bar(mapping = aes(x = seismoacoustic, fill=class)) +
  ylab('Count') + xlab('Hazard Assessment Level') +
  ggtitle('Seismoacoustic Method') + scale_fill_brewer(palette="Paired") +
  theme(text = element_text(size = font_size))

ghazard_barplot = ggplot(data = Seismic) + scale_x_discrete(drop=FALSE) +
  theme_minimal() + geom_bar(mapping = aes(x = ghazard, fill=class)) +
  ylab('Count') + xlab('Hazard Assessment Level') + ggtitle('Ghazard Method') +
  scale_fill_brewer(palette="Paired") +
  theme(text = element_text(size = font_size), legend.position="none")

shift_barplot = ggplot(data = Seismic) + scale_x_discrete(drop=FALSE) +
  geom_bar(mapping = aes(x = shift, fill=class)) + theme_minimal() +
  ylab('Count') + xlab('Type of Work') + ggtitle('Mining Shift') +
  scale_fill_brewer(palette="Paired") +
  theme(text = element_text(size = font_size), legend.position="none")

ggarrange(ggarrange(seismic_barplot, seismoacoustic_barplot, ncol = 2),
          ggarrange(ghazard_barplot, shift_barplot, ncol = 2), nrow = 2)
```



```
##############################    box plots    ##############################
SB_boxplot = Seismic %>% select(c(9:16), class) %>%
  pivot_longer(., cols = c(1:8), names_to = "SB",
               values_to = "Counts") %>%
  ggplot(aes(x = factor(SB),
             y = Counts, fill=class)) +
  geom_boxplot() + xlab('Seismic Bump Levels') +
  theme(text = element_text(size = font_size), legend.position="none",
        axis.title.x = element_text(vjust=-0.5)) +
  ggtitle('Seismic Bumps')

energy_boxplot = Seismic %>% select(c(4, 17, 18), class) %>%
  pivot_longer(., cols = c(1:3), names_to = "SB",
               values_to = "Counts") %>%
  ggplot(aes(x = factor(SB),
             y = Counts, fill=class)) +
```

```r
  geom_boxplot() + xlab('Energy Type') + ylab('Energy (J)') +
  theme(text = element_text(size = font_size),
        axis.title.x = element_text(vjust=-0.5)) +
  ggtitle('Energy')

dev_boxplot = Seismic %>% select(c(6,7), class) %>%
  pivot_longer(., cols = c(1:2), names_to = "SB",
               values_to = "Counts") %>%
  ggplot(aes(x = factor(SB),
             y = Counts, fill=class)) +
  geom_boxplot() + xlab('Type') + ylab('Deviance') +
  theme(text = element_text(size = font_size), legend.position="none",
        axis.title.x = element_text(vjust=-0.5)) +
  ggtitle('Deviance')

pulse_boxplot = Seismic %>% select(c(5), class) %>%
  pivot_longer(., cols = c(1), names_to = "SB",
               values_to = "Counts") %>%
  ggplot(aes(x = factor(SB),
             y = Counts, fill=class)) +
  geom_boxplot() + ylab('Counts') + xlab('Pulse')+
  theme(text = element_text(size = font_size), legend.position="none",
        axis.title.x = element_text(vjust=-0.5)) +
  ggtitle('Pulse')

ggarrange(ggarrange(SB_boxplot, energy_boxplot, ncol = 2),
          ggarrange(dev_boxplot, pulse_boxplot, ncol = 2), nrow = 2)
```



```r
# correlation
c = Seismic %>% select(c(4:7), c(9:13), c(17, 18))
round(cor(c), 2)
```

```
##             genergy gpuls gdenergy gdpuls nbumps nbumps2 nbumps3 nbumps4 nbumps5
## genergy        1.00  0.75     0.05   0.07   0.22    0.14    0.19    0.15   -0.01
## gpuls          0.75  1.00     0.29   0.38   0.30    0.21    0.23    0.26    0.05
## gdenergy       0.05  0.29     1.00   0.81   0.03    0.04   -0.01    0.04    0.12
## gdpuls         0.07  0.38     0.81   1.00   0.06    0.05    0.01    0.07    0.14
## nbumps         0.22  0.30     0.03   0.06   1.00    0.80    0.80    0.40    0.07
## nbumps2        0.14  0.21     0.04   0.05   0.80    1.00    0.35    0.16   -0.01
## nbumps3        0.19  0.23    -0.01   0.01   0.80    0.35    1.00    0.18    0.05
## nbumps4        0.15  0.26     0.04   0.07   0.40    0.16    0.18    1.00   -0.02
## nbumps5       -0.01  0.05     0.12   0.14   0.07   -0.01    0.05   -0.02    1.00
## energy         0.08  0.19     0.11   0.14   0.35    0.12    0.24    0.49    0.77
## maxenergy      0.06  0.16     0.11   0.14   0.27    0.09    0.18    0.42    0.81
##           energy maxenergy
## genergy     0.08      0.06
## gpuls       0.19      0.16
## gdenergy    0.11      0.11
## gdpuls      0.14      0.14
## nbumps      0.35      0.27
## nbumps2     0.12      0.09
## nbumps3     0.24      0.18
## nbumps4     0.49      0.42
## nbumps5     0.77      0.81
## energy      1.00      0.99
## maxenergy   0.99      1.00
```

```
# missing values
nrow(na.omit(Seismic)) - nrow(Seismic)
```

```
## [1] 0
```

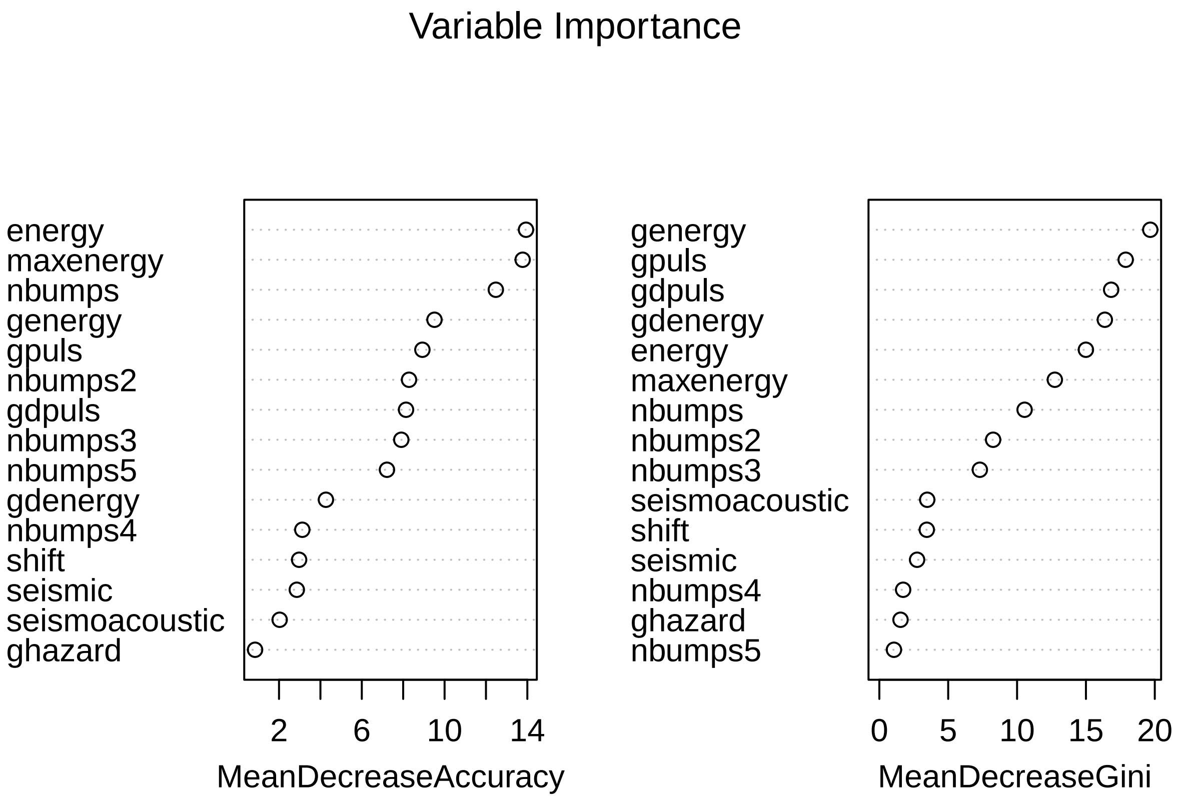
```
# drop columns and levels with zeros
Seismic = Seismic[-c(14, 15, 16)]
Seismic$seismic = factor(Seismic$seismic,
                         levels=c('a', 'b'))
Seismic$seismoacoustic = factor(Seismic$seismoacoustic,
                                levels=c('a', 'b', 'c'))
Seismic$ghazard = factor(Seismic$ghazard,
                         levels=c('a', 'b', 'c'))



################
#  data split  #
################


# subset of data such that 33% are examples
s0 = 2
```

```
s1 = Seismic[Seismic$class == 1,]

s2 = Seismic[Seismic$nbumps5 == 1 & Seismic$class == 0,]

s3 = Seismic[Seismic$nbumps5 == 0 & Seismic$class == 0,]

s4 = s3[sample(1:nrow(s3), 170*s0-11),]

Seismic_sub = rbind(s1, s2, s4)



# stratified 80/20 train-test split

index = createDataPartition(Seismic_sub$class, p = .8, list = FALSE)

train = Seismic_sub[index,]

test  = Seismic_sub[-index,]



###################

#  random forest  #

###################


# cross-validation

cv.rf = tune.randomForest(class ~ ., data = train,

                          mtry = 2:10,

                          ntree = 100*1:5,

                          tunecontrol=tune.control(sampling = "cross",cross=5))


# fit

rf = randomForest(class ~ ., data=train,

                  mtry=cv.rf$best.parameters$mtry,

                  ntree=cv.rf$best.parameters$ntree,

                  importance=TRUE,

                  type="class")


# optimal parameters
```

```r
cv.rf$best.parameters
```

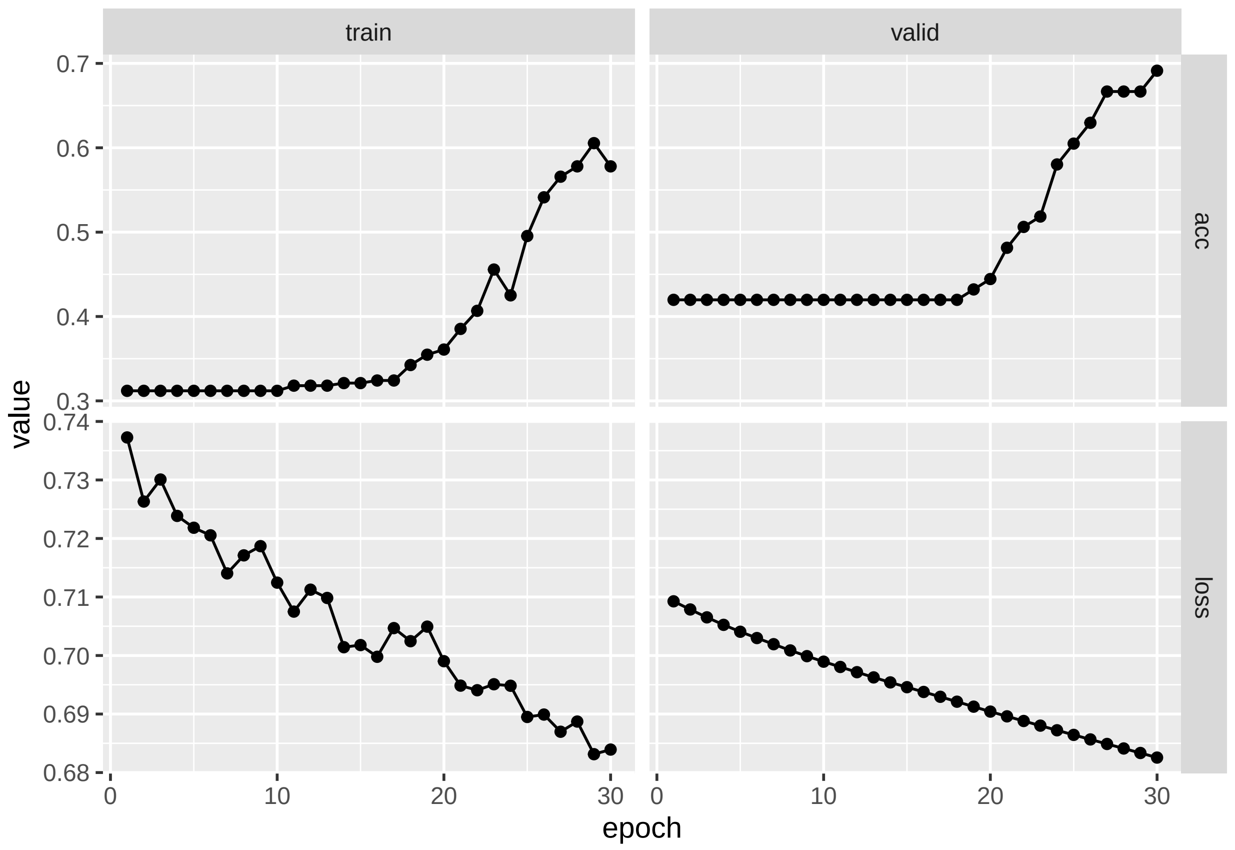
```
##    mtry ntree
## 28    2   400
```

```r
# variable importance
varImpPlot(rf, main="Variable Importance")
```



```r
####################
#  neural network  #
####################

# ensure matrix data has no NA values due to empty levels
z0 = as.data.frame(scale(model.matrix(class ~ . - 1, data = Seismic_sub)))
z1 = z0[ , colSums(is.na(z0))==0]

# convert matrix data to torch tensors
train_x = torch_tensor(as.matrix(z1[index,]))
```

```
train_y = torch_tensor(train$class)
test_x = torch_tensor(as.matrix(z1[-index,]))
test_y = torch_tensor(test$class)

# hyperparameters
isize = ncol(z1)
n1 = 10
n2 = 4
p1 = 0.25

# neural network model
nn = nn_module(
  initialize = function() {
    self$input = nn_linear(isize, n1)
    self$hidden1 = nn_linear(n1, n2)
    self$output = nn_linear(n2, 2)

    self$drop1 = nn_dropout(p1)

    self$activation = nn_relu()
    self$activation_out = nn_softmax(2)
  },
  forward = function(x) {
    x %>%

      self$input() %>%
      self$activation() %>%
      self$drop1() %>%

      self$hidden1() %>%
      self$activation() %>%
      self$output() %>%
      self$activation_out()
    })

# optimization
nn <- nn %>%
  setup(
    loss = nn_cross_entropy_loss(),
    optimizer = optim_adam,
    metrics = list(luz_metric_accuracy()))

# fit
fitted <- nn %>%
  fit(
      data = list(train_x, train_y),
      epochs = 30,
      valid_data = 0.2,
      dataloader_options = list(batch_size = 256),
      verbose = FALSE)

# training plot
plot(fitted)
```



```
################
#  performance  #
################


# performance function
performance = function(actual, predicted)
{
  t = table(actual, predicted)
  print(t)
  t_recall = t[2,2]/sum(t[2,])
  t_precision = t[2,2]/(sum(t[,2]))
  cat('Accuaracy:', sum(diag(t))/sum(t), '\n')
  cat('Sensitivity (SB):',t_recall, '\n')
  cat('Specificity (No SB):', t[1,1]/sum(t[1,]), '\n')
  cat('F1 Score:', (2*t_precision*t_recall)/(t_precision+t_recall), '\n')}
```

```
# cutoff function
cutoff = function(c, t, p)
{
  cat('Cutoff:', c)
  df = tibble(pre_prob = p, Y = t %>% pull(class))
  df = dplyr::mutate(df, Class_predicted = ifelse(pre_prob > c, 1, 0))
  return(df)}



# random forest performance
pred_forest = predict(rf, test, type="prob")[,2]
c_forest = cutoff(coords(roc(test$class, pred_forest), "best")$threshold,
                  test, pred_forest)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Cutoff: 0.39125
```

```
performance(c_forest$Y, c_forest$Class_predicted)
```

```
##       predicted
## actual  0  1
##      0 57 11
##      1 11 23
## Accuarcy: 0.7843137
## Sensitivity (SB): 0.6764706
## Specificity (No SB): 0.8382353
## F1 Score: 0.6764706
```

```
# neural network performance
n_p = as_array(predict(fitted, test_x))[,2]
c_nn = cutoff(coords(roc(test$class, n_p), "best")$threshold, test, n_p)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Cutoff: 0.5118262
```

```
performance(c_nn$Y, c_nn$Class_predicted)
```

```
##       predicted
## actual  0  1
##      0 55 13
##      1  8 26
## Accuarcy: 0.7941176
## Sensitivity (SB): 0.7647059
## Specificity (No SB): 0.8088235
## F1 Score: 0.7123288
```