STAT 8330 Project 1 Report: Training Part

Peng Shao, Wenyang Wang, Shuhua Zhang

November 16, 2015

Problem Description

Question 1 Part I:

This dataset summarizes a heterogeneous set of features about articles published by Mashable in a period of two years. The goal is to predict the logrithm of the number of shares in social networks.

Question 1 Part II:

The dataset is same to the dataset in part I, the goa is to classify the article into one of three levels of popularity, where `high` when shares more than 2100, `low` when shares no more than 1100 and `medium` for the rest.

Data Tidying and EDA

Since the dataset is very messy, we need to do some work to clean the data before the model training.

Basic Tidying Process

1. take the log of V59, saving into `news_reg` named `log_res`, and transform the V59 into factor varible, saving into `news_cls` named `class_res`, where <=1100 is low, >1100 and <=2100 is med, >2100 is high;
2. combine V12 to V17, create a factor variable called `type`, with 7 levels;
3. combine V30 to V36, create a factor variable called `weekday`, with 7 levels;
4. Transform V37 into factor;
5. Drop the columns: 12:17, 30:36, 59;
6. change the variables names.

Further Consideration

After basic transformation, there may still be so many missing values in different varibles.

1. The zero values of the number of words in contents

In fact, it is possible that a article can have no words. But the problem cause by the the zero values in variable `n_tokens_content` is that the values of `n_unique_tokens,n_non_stop_words,n_non_stop_unique_tokens` are all become zero and can be confusing, because any number from 0 to 1 is logically proper. It is necessary to take some acts to deal with this problem, so we decide four ways to manipulate the data set and create 8 new datasets(regression and classification for each):

1. No more actions

2. Remove the data with missing values
3. Separate the data into two parts, zero part and non-zero part
4. Let the missing values be average

2. Other Problems

Other problems, like the -1 values in the count data, or the high skewness of some variable, may have some influences on the model fitting, but we have no tenable reason that any kind of transformation, such as log or square root, will actually improve training precedure. Moreover, the more manipulation we do, the more biased the data will be, because we may put too much personal artificial information into the data. Thus, we decide to avoid more gratuitous transformation.

Exploratory Data Analysis

For the dataset 1(dataset for regression problem, without manipulation), We firstly try a linear regression based on all predictor. There are some coefficients being `NA`, which are usually caused by some obvious multicollinearity, like the dummy variables about weekdays. The most important thing is that the R_a^2 and R^2 is less than .2, which indicate that there is almost no linear relation between predictors and reponse. So we may guess the relation, no matter linear or nonlinear, is not so strong. If we plot the scatter plot of predicted values v.s. responses, the predictions are basiclly the mean value of responses with some noise.

Model Selection

Because:

1. There are so many features and the feature selection is a complete NP problem;
2. There are some variables completely relating to some of others.

According to these, random forest should be the best choice, if we do not want to do more transformation of data. The results of K. Fernandes, P. Vinagre and P. Cortez (2015) also show that the random forest performs better in two-level classfication problem.

For the four datasets above, we try the random forest model on them all, and tune the `mtry` option respectively. The results are:

1. The MSE of regression problem of data 1 is: `0.7234906`
2. The MSE of regression problem of data 2 is: `0.7141642`
3. The MSE of regression problem of data 3 is: `0.722908`
4. The MSE of regression problem of data 4 is: `0.724735`
5. The error rate of classfication problem of data 1 is: `0.4989333`
6. The error rate of classfication problem of data 2 is: `0.511`
7. The error rate of classfication problem of data 3 is: `0.4989667`
8. The error rate of classfication problem of data 4 is: `0.4990667`

We can see that, in fact, there is almost no difference between models. Specificly speaking, taking regression as an example, the difference among MSEs of logrithm of responses is only about 0.01, which means that the ratio between the prediction and response in original scale is about $e^{0.01} = 1.01$, i.e., 1% relative error. So the problems of data like missing values do not have so much influence on the model fitting. But, on the other hand, this may also indicate that the data is actually not that useful to make predictions.

Model Summary

We decide the model based on the first dataset (which is not manipulated except basic tidying process), i.e. the model `rf1r` and `rf1c` for regression and classification respectively. The tuning parameters `mtry` are 10 for regression and 9 for classfication.

The overall MSE for regression is as above: MSE=`0.7234906`.

The relevant results for classfication are:

```
confusionMatrix(news_cls1$class_res, rf1c$predicted)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  low  med high
##       low  7325 1517 2250
##       med  3634 2041 3357
##       high 2531 1679 5666
##
## Overall Statistics
##
##                Accuracy : 0.5011
##                  95% CI : (0.4954, 0.5067)
##     No Information Rate : 0.4497
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2411
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: low Class: med Class: high
## Sensitivity              0.5430    0.38973      0.5026
## Specificity              0.7718    0.71768      0.7752
## Pos Pred Value           0.6604    0.22597      0.5737
## Neg Pred Value           0.6739    0.84758      0.7214
## Prevalence               0.4497    0.17457      0.3758
## Detection Rate           0.2442    0.06803      0.1889
## Detection Prevalence     0.3697    0.30107      0.3292
## Balanced Accuracy        0.6574    0.55371      0.6389
```

The false positive rates are `1-Specificity`, which are all less than 30%, and for level `low` and level `high`, the false positive rates is less than 25%.

Conclusion

The random forest model for regression is almost useless, since the standard deviation of the `log_res` is only `0.9284743`. But the classification model is relatively useful, with the accuracy `50.11`%. Because this three-level classification, so random guess only can get `44.97`% accuracy.

Reference

K. Fernandes, P. Vinagre and P. Cortez. A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News. Proceedings of the 17th EPIA 2015 - Portuguese Conference on Artificial Intelligence, September, Coimbra, Portugal.

Problem Description

Question 2:

This dataset contains 700 observations of Hill-Valley pattern plots. Each record represents 100 points on a two-dimensional graph. When plotted in order (from 1 through 100) as the Y co-ordinate, the points will create either a Hill (labeled as 1) or a Valley (labeled as 0).

Model Description

This dataset is very special: most data are used to describe the terrain, and the noise is not large enough to suppress the top of the hill or the bottom of the valley, which means we can have a base line to detect the pattern of the plot.

We choose to normalize each observation, and to see whether the sign of largest absolute value of point is positive or not. If it is positive, it will be hill, otherwise it will be valley. The code is like

```
hill_valley <- function(hill) {
    norm_hill <- t(apply(as.matrix(hill)[, -101], 1, scale))
    return(ifelse(sign(norm_hill[cbind(1:700, apply(abs(norm_hill), 1, which.max))]) >
        0, 1, 0))
}
pred_hill <- hill_valley(hill)
```

Appendix (Code for Problem 1)

```
# R code for project 1
# Peng Shao, Wenyang Wang, Shuhua Zhang

# Reading data
news_raw <- read.table("./News_train.txt", header = FALSE)
hill <- read.table("./Hill-Valley_train.txt", header = FALSE)
news_names <- read.table("./names.txt", header = FALSE,
                         skip = 44, nrows = 59, sep = ":",
                         col.names = c("Name", "Description"))
# Data preprocessing
library(dplyr)
types <- c(sapply(strsplit(news_names[12:17, 1], "_"), "[", 4), "others")
weekdays <- sapply(strsplit(news_names[30:36, 1], "_"), "[", 3)
drops <- c(12:17, 30:36, 59)

news_reg <- news_raw %>%
```

```
        mutate(log_res = log(V59),
               type = factor(cbind(V12, V13, V14, V15, V16, V17,
                                   !(V12|V13|V14|V15|V16|V17)) %*%
                                     matrix(1:length(types)),
                             levels = 1:length(types),
                             labels = types),
               weekday = factor(cbind(V30, V31, V32, V33, V34, V35, V36) %*%
                                        matrix(1:length(weekdays)),
                                levels = 1:length(weekdays),
                                labels = weekdays),
               V37 = factor(V37, levels = c(1, 0), labels = c(TRUE, FALSE))) %>%
        select(-drops)
names(news_reg)[1:45] <- sapply(strsplit(news_names[, 1], ". "), "[", 2)[-drops]

news_cls <- news_raw %>%
        mutate(class_res = factor(cbind(V59<=1100, V59>1100&V59<=2100, V59>2100) %*%
                                          c(1, 2, 3),
                                  levels = 1:3,
                                  labels = c("low", "med", "high")),
               type = factor(cbind(V12, V13, V14, V15, V16, V17,
                                   !(V12|V13|V14|V15|V16|V17)) %*%
                                     matrix(1:length(types)),
                             levels = 1:length(types),
                             labels = types),
               weekday = factor(cbind(V30, V31, V32, V33, V34, V35, V36) %*%
                                        matrix(1:length(weekdays)),
                                levels = 1:length(weekdays),
                                labels = weekdays),
               V37 = factor(V37, levels = c(1, 0), labels = c(TRUE, FALSE))) %>%
        select(-drops)
names(news_cls)[1:45] <- sapply(strsplit(news_names[, 1], ". "), "[", 2)[-drops]

# Missing values processing
news_reg1 <- news_reg
news_cls1 <- news_cls

news_reg2 <- news_reg
news_reg2 <- news_reg2[!(news_reg2$n_tokens_content == 0), ]
news_cls2 <- news_cls
news_cls2 <- news_cls2[!(news_cls2$n_tokens_content == 0), ]

news_reg3 <- news_reg
news_cls3 <- news_cls
news_reg3_zero <- news_reg3[(news_reg3$n_tokens_content == 0), ]
news_reg3_nonzero <- news_reg3[!(news_reg3$n_tokens_content == 0), ]
news_cls3_zero <- news_cls3[(news_cls3$n_tokens_content == 0), ]
news_cls3_nonzero <- news_cls3[!(news_cls3$n_tokens_content == 0), ]

news_reg4 <- news_reg
news_cls4 <- news_cls
ind <- (news_reg4$n_tokens_content == 0)
news_reg4[ind, 2] <- round(mean(news_reg4[!ind, 2]))
news_reg4[ind, 3] <- mean(news_reg4[!ind, 3])
```

```
news_reg4[ind, 4] <- mean(news_reg4[!ind, 4])
news_reg4[ind, 5] <- mean(news_reg4[!ind, 5])
news_cls4[ind, 2] <- round(mean(news_cls4[!ind, 2]))
news_cls4[ind, 3] <- mean(news_cls4[!ind, 3])
news_cls4[ind, 4] <- mean(news_cls4[!ind, 4])
news_cls4[ind, 5] <- mean(news_cls4[!ind, 5])


# Model Fitting
# 1
input1r <- model.matrix(log_res ~., data = news_reg1)
output1r <- news_reg1$log_res
input1c <- model.matrix(class_res ~., data = news_cls1)
output1c <- news_cls1$class_res
ntree <- 250
# For regression
mse <- 10
rf1r <- randomForest(x = input1r[1:100, ], y = output1r[1:100],
                     mtry = 1, ntree = 1, do.trace = TRUE)
for (i in c(8:13)){
        print(i)
        rf_reg <- randomForest(x = input1r, y = output1r,
                               mtry = i, ntree = ntree, do.trace = TRUE)
        mse_new <- rf_reg$mse[ntree]
        if (mse_new < mse){
                mse <- mse_new
                rf1r <- rf_reg
        }
}
save(rf1r, file = "./rf1r.RData")
# For classification
err <- 1
rf1c <- randomForest(x = input1c[1:100, ], y = output1c[1:100],
                     mtry = 1, ntree = 1, do.trace = TRUE)
for (i in c(8:13)){
        print(i)
        rf_cls <- randomForest(x = input1c, y = output1c,
                               mtry = i, ntree = ntree, do.trace = TRUE)
        err_new <- rf_cls$err.rate[1]
        if (err_new < err){
                err <- err_new
                rf1c <- rf_cls
        }
}
save(rf1c, file = "./rf1c.RData")


# 2
input2r <- model.matrix(log_res ~., data = news_reg2)
output2r <- news_reg2$log_res
input2c <- model.matrix(class_res ~., data = news_cls2)
output2c <- news_cls2$class_res
ntree <- 250
```

```
# For regression
mse <- 10
rf2r <- randomForest(x = input2r[1:100, ], y = output2r[1:100],
                     mtry = 1, ntree = 1, do.trace = TRUE)
for (i in c(8:13)){
        print(i)
        rf_reg <- randomForest(x = input2r, y = output2r,
                               mtry = i, ntree = ntree, do.trace = TRUE)
        mse_new <- rf_reg$mse[ntree]
        if (mse_new < mse){
                mse <- mse_new
                rf2r <- rf_reg
        }
}
save(rf2r, file = "./rf2r.RData")
# For classification
err <- 1
rf2c <- randomForest(x = input2c[1:100, ], y = output2c[1:100],
                     mtry = 1, ntree = 1, do.trace = TRUE)
for (i in c(8:13)){
        print(i)
        rf_cls <- randomForest(x = input2c, y = output2c,
                               mtry = i, ntree = ntree, do.trace = TRUE)
        err_new <- rf_cls$err.rate[1]
        if (err_new < err){
                err <- err_new
                rf2c <- rf_cls
        }
}
save(rf2c, file = "./rf2c.RData")


# 3
input31 <- model.matrix(log_res ~., data = news_reg3_nonzero)
output31 <- news_reg3_nonzero$log_res
input32 <- model.matrix(log_res ~., data = news_reg3_zero)
output32 <- news_reg3_zero$log_res
input41 <- model.matrix(class_res ~., data = news_cls3_nonzero)
output41 <- news_cls3_nonzero$class_res
input42 <- model.matrix(class_res ~., data = news_cls3_zero)
output42 <- news_cls3_zero$class_res
ntree <- 250
# For regression
mse <- 10
rf31 <- randomForest(x = input31[1:100, ], y = output31[1:100],
                     mtry = 1, ntree = 1, do.trace = TRUE)
for (i in c(8:13)){
        print(i)
        rf_reg <- randomForest(x = input31, y = output31,
                               mtry = i, ntree = ntree, do.trace = TRUE)
        mse_new <- rf_reg$mse[ntree]
        if (mse_new < mse){
                mse <- mse_new
```

```
                rf31 <- rf_reg
        }
}
save(rf31, file = "./rf31.RData")
mse <- 10
rf32 <- randomForest(x = input32[1:100, ], y = output32[1:100],
                     mtry = 1, ntree = 1, do.trace = TRUE)
for (i in c(8:13)){
        print(i)
        rf_reg <- randomForest(x = input32, y = output32,
                               mtry = i, ntree = ntree, do.trace = TRUE)
        mse_new <- rf_reg$mse[ntree]
        if (mse_new < mse){
                mse <- mse_new
                rf31 <- rf_reg
        }
}
save(rf32, file = "./rf32.RData")
# for classification
err <- 1
rf41 <- randomForest(x = input41[1:100, ], y = output41[1:100],
                     mtry = 1, ntree = 1, do.trace = TRUE)
for (i in c(8:13)){
        print(i)
        rf_cls <- randomForest(x = input41, y = output41,
                               mtry = i, ntree = ntree, do.trace = TRUE)
        err_new <- rf_cls$err.rate[1]
        if (err_new < err){
                err <- err_new
                rf41 <- rf_cls
        }
}
save(rf41, file = "./rf41.RData")
err <- 1
rf42 <- randomForest(x = input42[1:100, ], y = output42[1:100],
                     mtry = 1, ntree = 1, do.trace = TRUE)
for (i in c(8:13)){
        print(i)
        rf_cls <- randomForest(x = input42, y = output42,
                               mtry = i, ntree = ntree, do.trace = TRUE)
        err_new <- rf_cls$err.rate[1]
        if (err_new < err){
                err <- err_new
                rf42 <- rf_cls
        }
}
save(rf42, file = "./rf42.RData")



# 4
input4r <- model.matrix(log_res ~., data = news_reg4)
output4r <- news_reg4$log_res
```

```
input4c <- model.matrix(class_res ~., data = news_cls4)
output4c <- news_cls4$class_res
ntree <- 250
mse <- 10
rf4r <- randomForest(x = input4r[1:100, ], y = output4r[1:100],
                     mtry = 1, ntree = 1, do.trace = TRUE)
for (i in c(8:13)){
        print(i)
        rf_reg <- randomForest(x = input4r, y = output4r,
                               mtry = i, ntree = ntree, do.trace = TRUE)
        mse_new <- rf_reg$mse[ntree]
        if (mse_new < mse){
                mse <- mse_new
                rf4r <- rf_reg
        }
}
save(rf4r, file = "./rf4r.RData")
# for classification
err <- 1
rf4c <- randomForest(x = input4c[1:100, ], y = output4c[1:100],
                     mtry = 1, ntree = 1, do.trace = TRUE)
for (i in c(8:13)){
        print(i)
        rf_cls <- randomForest(x = input4c, y = output4c,
                               mtry = i, ntree = ntree, do.trace = TRUE)
        err_new <- rf_cls$err.rate[1]
        if (err_new < err){
                err <- err_new
                rf4c <- rf_cls
        }
}
save(rf4c, file = "./rf4c.RData")
```