homework4

syh

May 31, 2017

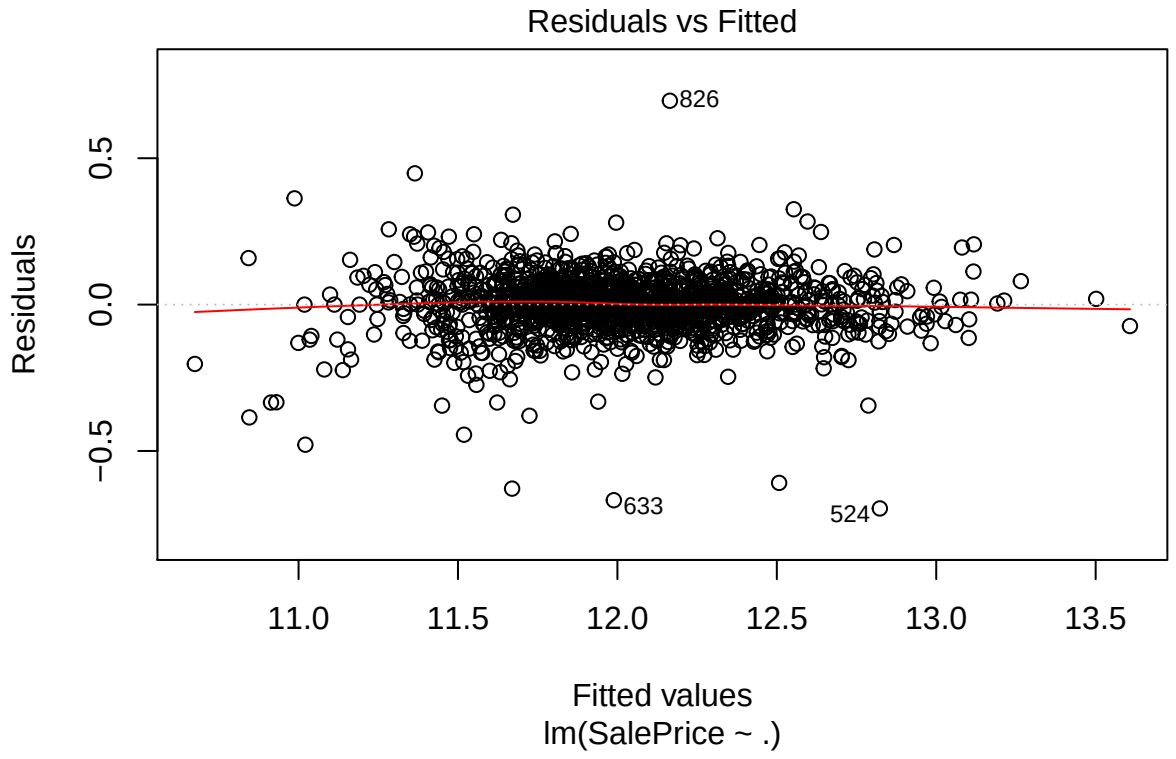
```
# in this section, we try to promote our linear regression model by dropping off outliers
```

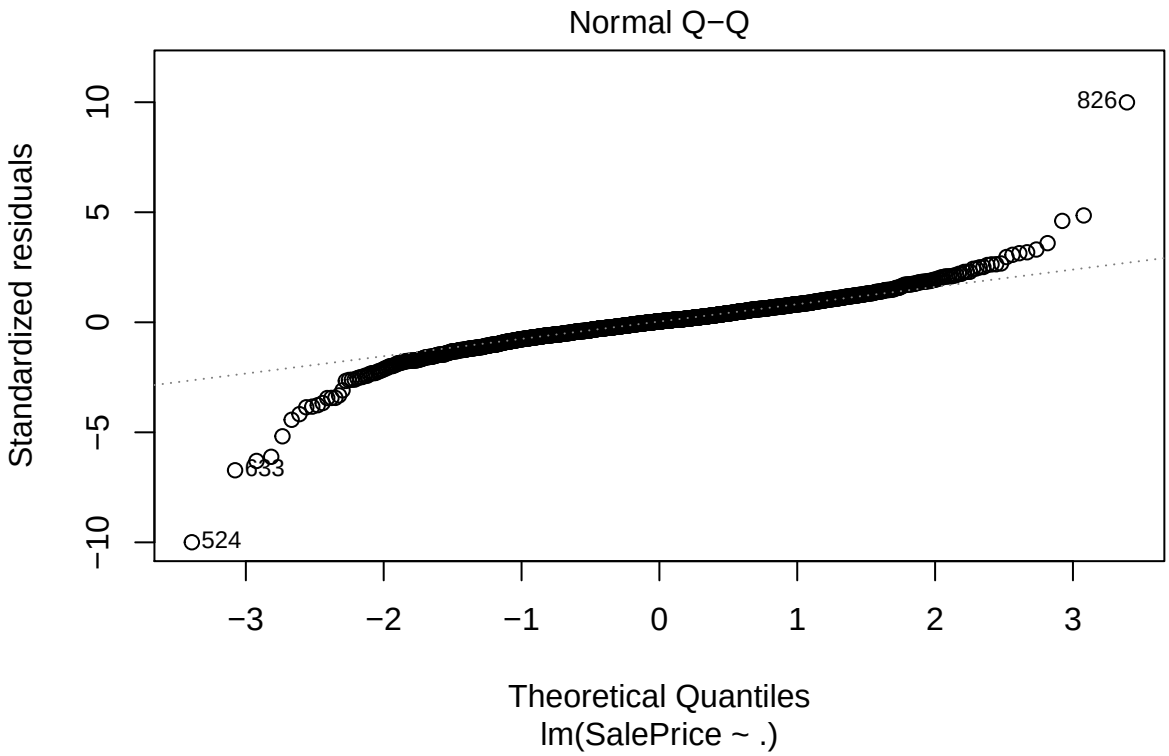
```
# read all data (train + prediction) without missing value
real_all_daat <- read.csv(file = "H:/kaggle/houseprice/data/real_all_data_hybrid.csv",
                          stringsAsFactors = FALSE)[,-c(1,2)]
```

```
# transform sale price to log sale price
real_all_daat[,"SalePrice"] <- log(real_all_daat[,"SalePrice"])
```

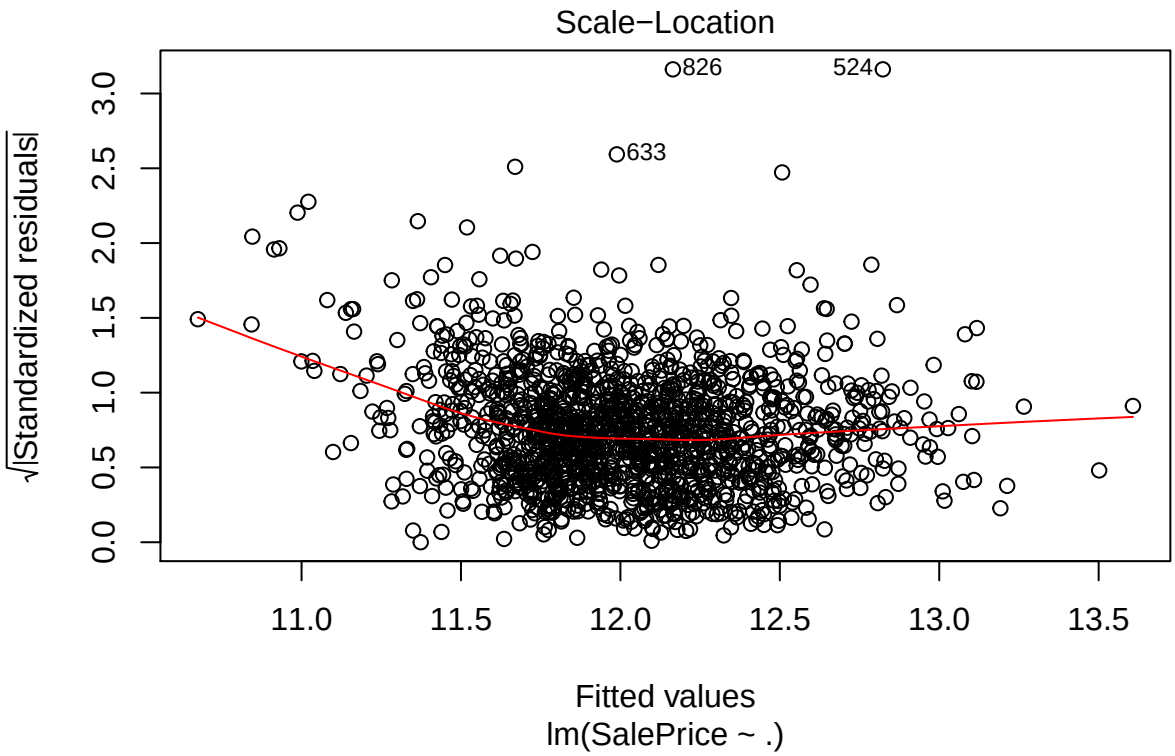
```
# train a simple linear model by lm()
# check if it's subject to linear regression
# find some outlier and delete them
simple_lm <- lm(formula = SalePrice ~., data = real_all_daat[1:1460,])
plot(simple_lm)
```

```
## Warning: not plotting observations with leverage one:
##   121, 251, 272, 333, 376, 399, 584, 596, 667, 945, 949, 1004, 1012, 1188, 1231, 1271, 1276, 1299, 13
```

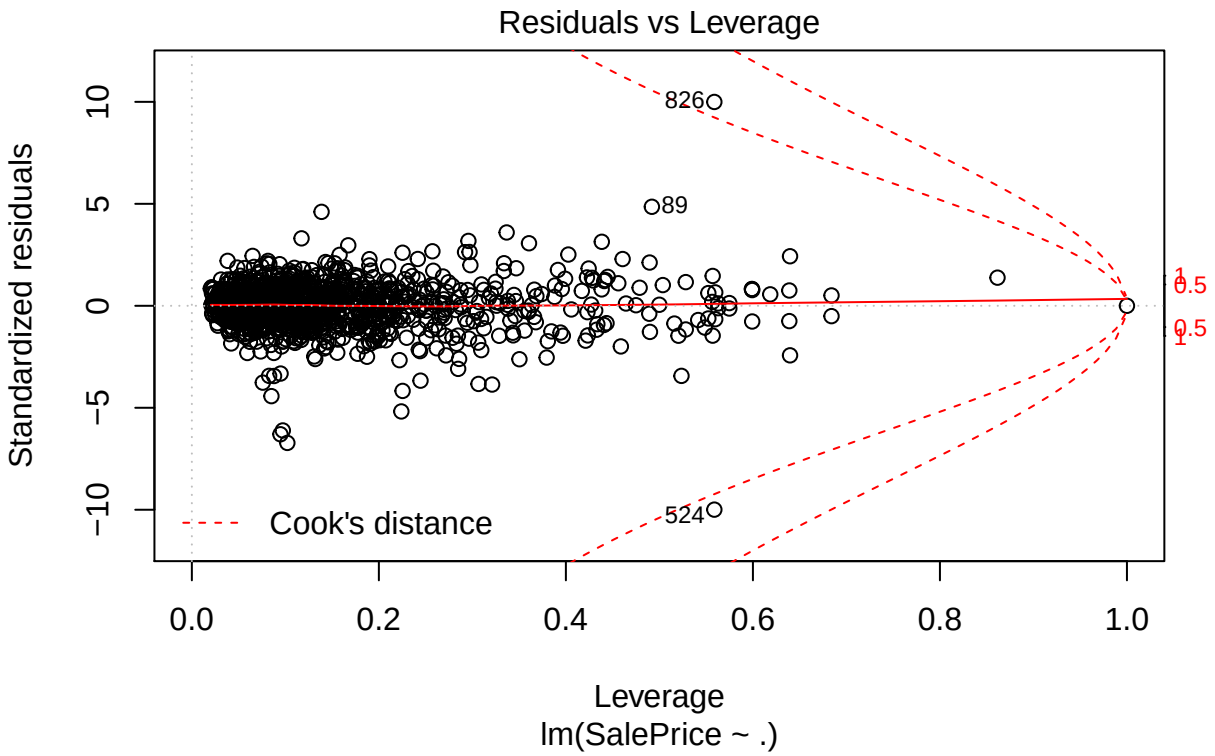





```
## Warning: not plotting observations with leverage one:
##   121, 251, 272, 333, 376, 399, 584, 596, 667, 945, 949, 1004, 1012, 1188, 1231, 1271, 1276, 1299, 13
```




```
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```




```
# from above, we know basically it can suject to linear regression
# there are some outliers: 89, 524, 633, 826
# get rid of them now after converting data
```

```
# we would like to train a linear regression model with regulation
# 1. convert categorical ones to factors
for(i in 1:dim(real_all_daat)[2]){
  if(is.character(real_all_daat[,i])){
    real_all_daat[,i] <- as.factor(real_all_daat[,i])
  }
}
```

```
# 2. convert x to matrix
all_data_x_matrix <- model.matrix(~.-SalePrice, data = real_all_daat)
```

```
# 1. split all data into train and prediction
model_x <- all_data_x_matrix[1:1460,]
model_y <- real_all_daat[1:1460,"SalePrice"]

pre_x <- all_data_x_matrix[-c(1:1460),]

# 2. get rid of outlier in train data
model_x <- model_x[-c(89, 524, 633, 826),]
model_y <- model_y[-c(89, 524, 633, 826)]

# 3. split model data into train and test
# set.seed(1000)
```

```
# train_ind <- sample(1:dim(model_x), size = dim(model_x) * 0.7)
train_ind <- read.csv(file = "H:/kaggle/houseprice/data/train_index.csv")
train_ind <- train_ind[,1]

train_x <- model_x[train_ind,]
train_y <- model_y[train_ind]

test_x <- model_x[-train_ind,]
test_y <- model_y[-train_ind]
```

```
# use cross validation to find optimal alpha in penality
library(glmnet)
```

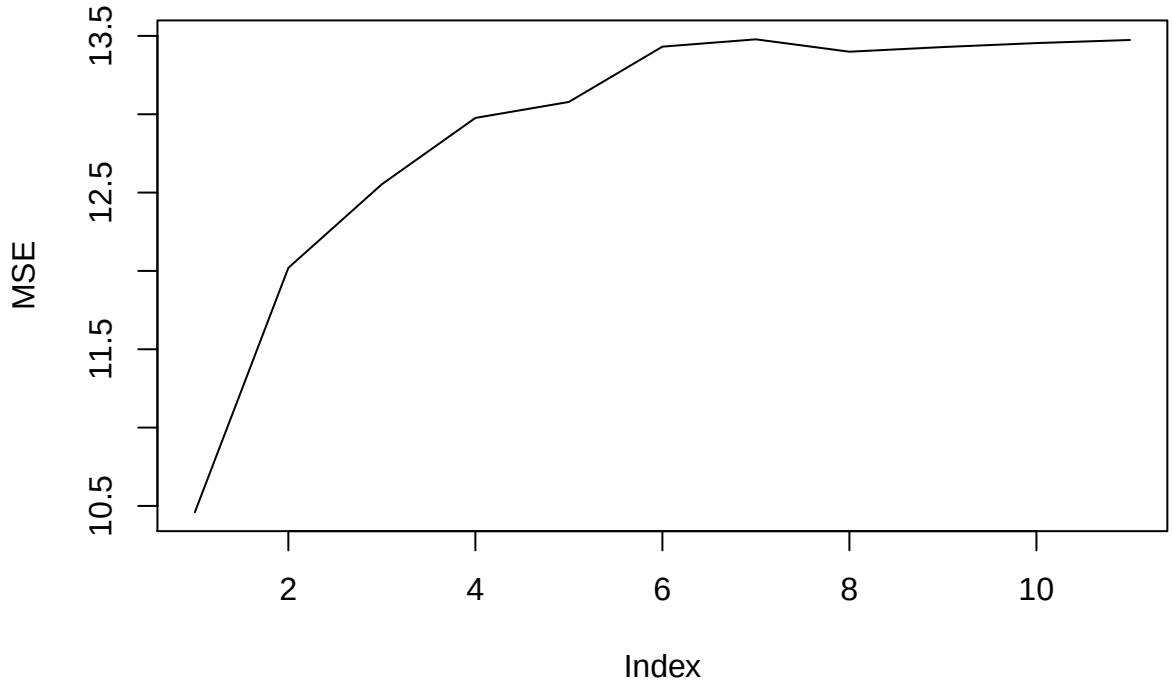
```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-10
```

```
for(i in 0:10){
  assign(paste("fit", i, sep = ""),
         cv.glmnet(x = train_x, y = train_y, alpha = i/10,
            type.measure = "mse", family = "gaussian"))
}
```

```
# find the optimal fit with least sse on test data
fit <- list(fit0, fit1, fit2, fit3, fit4, fit5, fit6, fit7, fit8, fit9, fit10)
mse <- NULL
for(i in 1:11){
  pre_y <- predict.cv.glmnet(object = fit[[i]], newx = test_x,s = fit[[i]]$lambda.1se)
  tmp_mse <- sum((pre_y - test_y) ^ 2)
  mse <- c(mse, tmp_mse)
}
plot(mse,type = 'l', ylab = "MSE")
```




```
# let's use this model to make prediction
opt_ind <- which.min(mse)
log_pre_y <- predict.cv.glmnet(object = fit[[opt_ind]],
                               newx = pre_x,s = fit[[opt_ind]]$lambda.1se)

# create submission
result <- data.frame(Id = c(1461:2919), SalePrice = exp(log_pre_y))
names(result) <- c("Id", "SalePrice")
write.csv(x = result, file = "H:/kaggle/houseprice/data/submission_3.csv",
          row.names = FALSE)
# on kaggle, error is 0.13464, promoted from last one
```