Regression Case Study Code

Guoqiang Liang, Fang Wang, Bingyi Li

10/6/2017

```
housing.data <- read.csv("/Users/asen/USF/601 Regression/Housing_Study/housing.txt")
housing.data %<>% select(-c(Id))
housing.morty <- read.csv("/Users/asen/USF/601 Regression/Housing_Study/morty.txt")
housing.morty %<>% select(-c(Id, X))
housing.data <- rbind(housing.data, housing.morty)

# change MSSubClass into factor
housing.data$MSSubClass %<>% as.factor()

# Replace missing data in LotFrontage with average
housing.data$LotFrontage[which(is.na(housing.data$LotFrontage))] <- mean(housing.data$LotFrontage,
    na.rm = TRUE)

# Replace missing data in Alley with None
housing.data$Alley <- as.character(housing.data$Alley)
housing.data$Alley[which(is.na(housing.data$Alley))] <- "None"
housing.data$Alley %<>% as.factor()

# Refactor YearBuilt to be the age of the building
housing.data$YearBuilt <- 2017 - housing.data$YearBuilt

# Refactor YearRemodAdd to be the number of years after
# last modification
housing.data$YearRemodAdd <- 2017 - housing.data$YearRemodAdd

# Replace missing data in MasVnrType with None
housing.data$MasVnrType <- as.character(housing.data$MasVnrType)
housing.data$MasVnrType[which(is.na(housing.data$MasVnrType))] <- "None"
housing.data$MasVnrType %<>% as.factor()

# Replace missing data in MasVnrArea with 0
housing.data$MasVnrArea[which(is.na(housing.data$MasVnrArea))] <- 0

# Change ExterQual to integers
housing.data$ExterQual <- as.numeric(factor(housing.data$ExterQual,
    levels = c("Po", "Fa", "TA", "Gd", "Ex"), ordered = TRUE))

# Change ExterCond to integers
housing.data$ExterCond <- as.numeric(factor(housing.data$ExterCond,
    levels = c("Po", "Fa", "TA", "Gd", "Ex"), ordered = TRUE))

# Change BsmtQual to integers
housing.data$BsmtQual <- as.character(housing.data$BsmtQual)
housing.data$BsmtQual[which(is.na(housing.data$BsmtQual))] <- "None"
housing.data$BsmtQual <- as.numeric(factor(housing.data$BsmtQual,
    levels = c("None", "Po", "Fa", "TA", "Gd", "Ex"), ordered = TRUE))
```

```r
# Change BsmtCond to integers
housing.data$BsmtCond <- as.character(housing.data$BsmtCond)
housing.data$BsmtCond[which(is.na(housing.data$BsmtCond))] <- "None"
housing.data$BsmtCond <- as.numeric(factor(housing.data$BsmtCond,
    levels = c("None", "Po", "Fa", "TA", "Gd", "Ex"), ordered = TRUE))

# Change BsmtExposure to integers
housing.data$BsmtExposure <- as.character(housing.data$BsmtExposure)
housing.data$BsmtExposure[which(is.na(housing.data$BsmtExposure))] <- "None"
housing.data$BsmtExposure <- as.numeric(factor(housing.data$BsmtExposure,
    levels = c("None", "No", "Mn", "Av", "Gd"), ordered = TRUE))

# Replace missing data in BsmtFinType1 with None
housing.data$BsmtFinType1 <- as.character(housing.data$BsmtFinType1)
housing.data$BsmtFinType1[which(is.na(housing.data$BsmtFinType1))] <- "None"
housing.data$BsmtFinType1 %<>% as.factor()

# Replace missing data in BsmtFinType2 with None
housing.data$BsmtFinType2 <- as.character(housing.data$BsmtFinType2)
housing.data$BsmtFinType2[which(is.na(housing.data$BsmtFinType2))] <- "None"
housing.data$BsmtFinType2 %<>% as.factor()

# Change HeatingQC to integers
housing.data$HeatingQC <- as.numeric(factor(housing.data$HeatingQC,
    levels = c("Po", "Fa", "TA", "Gd", "Ex"), ordered = TRUE))

# Change KitchenQual to integers
housing.data$KitchenQual <- as.numeric(factor(housing.data$KitchenQual,
    levels = c("Po", "Fa", "TA", "Gd", "Ex"), ordered = TRUE))

# Change Functional to integers
housing.data$Functional <- as.numeric(factor(housing.data$Functional,
    levels = c("Sal", "Sev", "Maj2", "Maj1", "Mod", "Min2",
        "Min1", "Typ"), ordered = TRUE))

# Change FireplaceQu to integers
housing.data$FireplaceQu <- as.character(housing.data$FireplaceQu)
housing.data$FireplaceQu[which(is.na(housing.data$FireplaceQu))] <- "None"
housing.data$FireplaceQu <- as.numeric(factor(housing.data$FireplaceQu,
    levels = c("None", "Po", "Fa", "TA", "Gd", "Ex"), ordered = TRUE))

# Replace missing data in GarageType with None
housing.data$GarageType <- as.character(housing.data$GarageType)
housing.data$GarageType[which(is.na(housing.data$GarageType))] <- "None"
housing.data$GarageType %<>% as.factor()

# Refactor GarageYrBlt to be the number of years after
# last modification
housing.data$GarageYrBlt <- 2017 - housing.data$GarageYrBlt
housing.data$GarageYrBlt[which(is.na(housing.data$GarageYrBlt))] <- mean(housing.data$GarageYrBlt,
    na.rm = TRUE)

# Change GarageFinish to integers
```

```r
housing.data$GarageFinish <- as.character(housing.data$GarageFinish)
housing.data$GarageFinish[which(is.na(housing.data$GarageFinish))] <- "None"
housing.data$GarageFinish <- as.numeric(factor(housing.data$GarageFinish,
    levels = c("None", "Unf", "RFn", "Fin"), ordered = TRUE))

# Change GarageQual to integers
housing.data$GarageQual <- as.character(housing.data$GarageQual)
housing.data$GarageQual[which(is.na(housing.data$GarageQual))] <- "None"
housing.data$GarageQual <- as.numeric(factor(housing.data$GarageQual,
    levels = c("None", "Po", "Fa", "TA", "Gd", "Ex"), ordered = TRUE))

# Change GarageCond to integers
housing.data$GarageCond <- as.character(housing.data$GarageCond)
housing.data$GarageCond[which(is.na(housing.data$GarageCond))] <- "None"
housing.data$GarageCond <- as.numeric(factor(housing.data$GarageCond,
    levels = c("None", "Po", "Fa", "TA", "Gd", "Ex"), ordered = TRUE))

# Change PavedDrive to integers
housing.data$PavedDrive <- as.numeric(factor(housing.data$PavedDrive,
    levels = c("N", "P", "Y"), ordered = TRUE))

# Change PoolQC to integers
housing.data$PoolQC <- as.character(housing.data$PoolQC)
housing.data$PoolQC[which(is.na(housing.data$PoolQC))] <- "None"
housing.data$PoolQC <- as.numeric(factor(housing.data$PoolQC,
    levels = c("None", "Fa", "TA", "Gd", "Ex"), ordered = TRUE))

# Change Fence to integers
housing.data$Fence <- as.character(housing.data$Fence)
housing.data$Fence[which(is.na(housing.data$Fence))] <- "None"
housing.data$Fence <- as.numeric(factor(housing.data$Fence,
    levels = c("None", "MnWw", "GdWo", "MnPrv", "GdPrv"),
    ordered = TRUE))

# Replace missing data in MiscFeature with None
housing.data$MiscFeature <- as.character(housing.data$MiscFeature)
housing.data$MiscFeature[which(is.na(housing.data$MiscFeature))] <- "None"
housing.data$MiscFeature %<>% as.factor()

# Refactor YrSold to be the length of time
housing.data$YrSold <- 2017 - housing.data$YrSold

# drop the row missing 'Electrical'
housing.data %<>% na.omit()

#------log area------#
# bofore log
plot(density(log(housing.data$LotArea + 1)))
# after log
plot(density(housing.data$LotArea))

housing.data$LotArea <- log(housing.data$LotArea + 1)
# hist(housing.data$LotArea)
```

```
housing.data$MasVnrArea <- log(housing.data$MasVnrArea +
    1)
# hist(housing.data$MasVnrArea)

housing.data$X1stFlrSF <- log(housing.data$X1stFlrSF + 1)
# hist(housing.data$X1stFlrSF)

housing.data$GrLivArea <- log(housing.data$GrLivArea + 1)
# hist(housing.data$GrLivArea)

#------end of log------#

#------condition/exterior------#
temp = c("Artery", "Feedr", "RRNn", "RRAn", "PosN", "PosA",
    "RRNe", "RRAe")

for (j in 1:length(temp)) {
    housing.data[[temp[j]]] = 0
}

for (i in 1:length(housing.data[, 1])) {
    for (j in 1:length(temp)) {
        if (housing.data$Condition1[i] == temp[j] | housing.data$Condition2[i] ==
            temp[j]) {
            housing.data[[temp[j]]][i] = 1
        }
    }
}

temp = c("AsbShng", "AsphShn", "BrkComm", "BrkFace", "CBlock",
    "CemntBd", "HdBoard", "ImStucc", "MetalSd", "Plywood",
    "PreCast", "Stone", "Stucco", "VinylSd", "Wd Sdng",
    "WdShing")

for (j in 1:length(temp)) {
    housing.data[[temp[j]]] = 0
}

for (i in 1:length(housing.data[, 1])) {
    for (j in 1:length(temp)) {
        if (housing.data$Exterior1st[i] == temp[j] | housing.data$Exterior2nd[i] ==
            temp[j]) {
            housing.data[[temp[j]]][i] = 1
        }
    }
}

housing.data %<>% select(-c(Condition1, Condition2, Exterior1st,
    Exterior2nd, PreCast))
housing.morty <- housing.data[nrow(housing.data), ]
housing.data <- housing.data[-nrow(housing.data), ]
#------end condition/exterior------#
```

```r
#------scale------#
for (i in 1:length(housing.data)) {
    # print(class(housing.data[[i]]))
    if (colnames(housing.data[i]) != "SalePrice") {
        if (class(housing.data[[i]]) == "numeric" | class(housing.data[[i]]) ==
            "integer") {
            if (max(housing.data[[i]]) != 1 | min(housing.data[[i]]) !=
                0) {
                housing.data[[i]] = scale(housing.data[[i]])
            }
        }
    }
}
#------end of scale------#
```

```r
#------backward selction------#

housing.data %<>% dplyr::select(-c(CBlock, Utilities))

full.model <- lm(SalePrice ~ ., data = housing.data)
null.model <- lm(SalePrice ~ 1, data = housing.data)

backwards <- step(full.model, data = housing.data, direction = "both",
    k = log(1459))
length(backwards$coefficients)
Mse_2 <- sqrt(mean((backwards$fitted.values - housing.data$SalePrice)^2))
sqrt(sum((backwards$fitted.values - housing.data$SalePrice)^2)/1328)
Mse_2

summary(backwards)

###### ols #####

backwards <- lm(SalePrice ~ LotArea + Street + LotShape +
    LandContour + LotConfig + Neighborhood + BldgType +
    OverallQual + OverallCond + YearBuilt + RoofMatl + MasVnrArea +
    ExterQual + ExterCond + BsmtQual + BsmtExposure + BsmtFinType1 +
    BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF + X1stFlrSF + X2ndFlrSF +
    GrLivArea + FullBath + HalfBath + BedroomAbvGr + KitchenAbvGr +
    KitchenQual + TotRmsAbvGrd + Functional + Fireplaces +
    FireplaceQu + GarageType + GarageCars + GarageArea +
    GarageQual + WoodDeckSF + X3SsnPorch + ScreenPorch +
    PoolQC + Fence + MiscFeature + MoSold + SaleCondition +
    Artery + Feedr + PosN + RRNe + RRAe + BrkFace + ImStucc +
    MetalSd + Plywood, data = housing.data)

library(olsrr)
ols_dffits_plot(backwards)

dffits_bs <- abs(dffits(backwards))
dffits_bs[is.na(dffits(backwards))] <- 10

x_new <- housing.data %>% dplyr::select(-c(SalePrice))
```

```
dffits_data_bs <- housing.data[dffits_bs < 2 * sqrt(ncol(x_new)/nrow(x_new)),
    ]

backwards_dffits <- lm(SalePrice ~ LotArea + Street + LotShape +
    LandContour + LotConfig + Neighborhood + BldgType +
    OverallQual + OverallCond + YearBuilt + RoofMatl + MasVnrArea +
    ExterQual + ExterCond + BsmtQual + BsmtExposure + BsmtFinType1 +
    BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF + X1stFlrSF + X2ndFlrSF +
    GrLivArea + FullBath + HalfBath + BedroomAbvGr + KitchenAbvGr +
    KitchenQual + TotRmsAbvGrd + Functional + Fireplaces +
    FireplaceQu + GarageType + GarageCars + GarageArea +
    GarageQual + WoodDeckSF + X3SsnPorch + ScreenPorch +
    PoolQC + Fence + MiscFeature + MoSold + SaleCondition +
    Artery + Feedr + PosN + RRNe + RRAe + BrkFace + ImStucc +
    MetalSd + Plywood, data = dffits_data_bs)


e_bs_dffits <- backwards_dffits$residuals
std.resi_bs_dffits <- (e_bs_dffits - mean(e_bs_dffits))/sd(e_bs_dffits)
ks.test(std.resi_bs_dffits, rnorm(1000))

sum_ks_ols <- 0
for (i in 1:100) {
    sum_ks_ols <- sum_ks_ols + ks.test(std.resi_dffits_sqrt,
        rnorm(1000))[2][[1]]
}
sum_ks_ols/100

###### end of ols #####



library(MASS)
boxcox(backwards_dffits)

###### log transfromation #####

backwards_log <- lm(log(SalePrice) ~ LotArea + Street +
    LotShape + LandContour + LotConfig + Neighborhood +
    BldgType + OverallQual + OverallCond + YearBuilt + RoofMatl +
    MasVnrArea + ExterQual + ExterCond + BsmtQual + BsmtExposure +
    BsmtFinType1 + BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF +
    X1stFlrSF + X2ndFlrSF + GrLivArea + FullBath + HalfBath +
    BedroomAbvGr + KitchenAbvGr + KitchenQual + TotRmsAbvGrd +
    Functional + Fireplaces + FireplaceQu + GarageType +
    GarageCars + GarageArea + GarageQual + WoodDeckSF +
    X3SsnPorch + ScreenPorch + PoolQC + Fence + MiscFeature +
    MoSold + SaleCondition + Artery + Feedr + PosN + RRNe +
    RRAe + BrkFace + ImStucc + MetalSd + Plywood, data = housing.data)

e_log <- backwards_log$residuals
std.resi_log <- (e_log - mean(e_log))/sd(e_log)
ks.test(std.resi_log, rnorm(1000))
```

```r
ols_dffits_plot(backwards_log)

dffits_bs_log <- abs(dffits(backwards_log))
dffits_bs_log[is.na(dffits(backwards_log))] <- 10

x_new <- housing.data %>% dplyr::select(-c(SalePrice))

dffits_data_bs_log <- housing.data[dffits_bs_log < 2 * sqrt(ncol(x_new)/nrow(x_new)),
    ]

backwards_log_dffits <- lm(log(SalePrice) ~ LotArea + Street +
    LotShape + LandContour + LotConfig + Neighborhood +
    BldgType + OverallQual + OverallCond + YearBuilt + RoofMatl +
    MasVnrArea + ExterQual + ExterCond + BsmtQual + BsmtExposure +
    BsmtFinType1 + BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF +
    X1stFlrSF + X2ndFlrSF + GrLivArea + FullBath + HalfBath +
    BedroomAbvGr + KitchenAbvGr + KitchenQual + TotRmsAbvGrd +
    Functional + Fireplaces + FireplaceQu + GarageType +
    GarageCars + GarageArea + GarageQual + WoodDeckSF +
    X3SsnPorch + ScreenPorch + PoolQC + Fence + MiscFeature +
    MoSold + SaleCondition + Artery + Feedr + PosN + RRNe +
    RRAe + BrkFace + ImStucc + MetalSd + Plywood, data = dffits_data_bs_log)



e_dffits_log <- backwards_log_dffits$residuals
std.resi_dffits_log <- (e_dffits_log - mean(e_dffits_log))/sd(e_dffits_log)

sum_ks <- 0
for (i in 1:100) {
    sum_ks <- sum_ks + ks.test(std.resi_dffits_log, rnorm(1000))[2][[1]]
}
sum_ks/100

ks.test(std.resi_dffits_log, rnorm(1000))
shapiro.test(std.resi_dffits_log)

library(faraway)
vif(backwards_log_dffits)

par(mfrow = c(2, 2))
plot(backwards_log_dffits, add.smooth = F)

###### end of log transfromation #####

predict(backwards_log_dffits, housing.morty, interval = "predict")



#### lasso feature selection ####

x <- model.matrix(SalePrice ~ ., data = housing.data)[,
```

```
    -1]
y <- housing.data$SalePrice

set.seed(100)
cv.out <- cv.glmnet(x, y, alpha = 1)
best.lambda <- cv.out$lambda.min
plot(cv.out)
abline(v = log(best.lambda), col = "blue", lwd = 2)
best.lambda
lasso2_cv <- glmnet(x, y, alpha = 1, lambda = best.lambda)
coef_lasso <- coef(lasso2_cv)

coef_matrix <- as.matrix(coef_lasso)
coef_df <- data.frame(feature = rownames(coef_matrix), d1 = coef_matrix[,
    1])
coef_df %<>% filter(d1 != 0 & feature != "(Intercept)")

selected_x <- data.frame(x) %>% dplyr::select(coef_df$feature)
selected_data <- cbind(selected_x, SalePrice = y)


#### ols ####
lm.model.lasso <- lm(SalePrice ~ ., data = selected_data)

ols_dffits_plot(lm.model.lasso)

dffits_lasso <- abs(dffits(lm.model.lasso))
dffits_lasso[is.na(dffits(lm.model.lasso))] <- 10

x_new <- housing.data %>% dplyr::select(-c(SalePrice))

dffits_data_lasso <- selected_data[dffits_lasso < 2 * sqrt(ncol(x_new)/nrow(x_new)),
    ]

lasso_dffits <- lm(SalePrice ~ ., data = dffits_data_lasso)


e_lasso_dffits <- lasso_dffits$residuals
std.resi_lasso_dffits <- (e_lasso_dffits - mean(e_lasso_dffits))/sd(e_lasso_dffits)
ks.test(std.resi_lasso_dffits, rnorm(1000))

boxcox(lasso_dffits)

sum_ks_lasso <- 0
for (i in 1:100) {
    sum_ks_lasso <- sum_ks_lasso + ks.test(std.resi_lasso_dffits,
        rnorm(1000))[2][[1]]
}
sum_ks_lasso/100
### end ols ###


### log transformation ###
```

```r
lm.model.lasso.log <- lm(log(SalePrice) ~ ., data = selected_data)

ols_dffits_plot(lm.model.lasso.log)

dffits_lasso.log <- abs(dffits(lm.model.lasso.log))
dffits_lasso.log[is.na(dffits(lm.model.lasso.log))] <- 10

x_new <- housing.data %>% dplyr::select(-c(SalePrice))

dffits_data_lasso.log <- selected_data[dffits_lasso.log <
    2 * sqrt(ncol(x_new)/nrow(x_new)), ]

lasso_dffits.log <- lm(SalePrice ~ ., data = dffits_data_lasso.log)


e_lasso_dffits.log <- lasso_dffits.log$residuals
std.resi_lasso_dffits.log <- (e_lasso_dffits.log - mean(e_lasso_dffits.log))/sd(e_lasso_dffits.log)
ks.test(std.resi_lasso_dffits.log, rnorm(1000))



sum_ks_lasso.log <- 0
for (i in 1:100) {
    sum_ks_lasso.log <- sum_ks_lasso.log + ks.test(std.resi_lasso_dffits.log,
        rnorm(1000))[2][[1]]
}
sum_ks_lasso.log/100

par(mfrow = c(2, 2))
plot(lasso_dffits.log)
```

```r
#------data split------#
set.seed(412)
rows <- dim(housing.data)[1]
training_count <- as.integer(rows * 0.7)
indices <- sample(1:rows, rows, replace = FALSE)
x_total <- model.matrix(SalePrice ~ ., data = housing.data)[,
    -1]
y_total <- log(housing.data$SalePrice)
training_x <- x_total[indices[1:training_count], ]
training_y <- y_total[indices[1:training_count]]
test_x <- x_total[-indices[1:training_count], ]
test_y <- y_total[-indices[1:training_count]]
#------end data split------#

cols.remove <- which(apply(training_x, 2, var) == 0)
training_x <- training_x[, -cols.remove]
test_x <- test_x[, -cols.remove]


pca_scale <- prcomp(training_x, center = TRUE, scale. = TRUE)

#------scale------#
training_x %<>% as.data.frame()
```

```r
for (i in 1:length(training_x)) {
    if (class(training_x[[i]]) == "numeric" | class(training_x[[i]]) ==
        "integer") {
        if (max(training_x[[i]]) != 1 | min(training_x[[i]]) !=
            0) {
            training_x[[i]] = scale(training_x[[i]])
        }
    }
}
training_x %<>% as.matrix()
#------end of scale------#

pca <- prcomp(training_x, center = FALSE, scale. = FALSE)
train.eigs <- pca$sdev^2
sum(train.eigs[1:54])/sum(train.eigs)

par(mfrow = c(1, 2))
x.pvar <- (pca$sdev^2)/sum(pca$sdev^2)
# barplot(x.pvar,ylim=c(0,0.1),xlab='Components',ylab='proportion
# of variance') cumulated varianced explained
plot(cumsum(x.pvar), ylim = c(0, 1), type = "b", xlab = "Number of components",
    ylab = "cumulated varianced")
# Generate the scree plot
screeplot(pca, type = "l", npcs = 100)


training_x <- pca$x[, 1:54]
#------scale------#
test_x %<>% as.data.frame()
for (i in 1:length(test_x)) {
    if (class(test_x[[i]]) == "numeric" | class(test_x[[i]]) ==
        "integer") {
        if (max(test_x[[i]]) != 1 | min(test_x[[i]]) !=
            0) {
            test_x[[i]] = scale(test_x[[i]], pca_scale$center[[i]],
                pca_scale$scale[[i]])
        }
    }
}
test_x %<>% as.matrix()
#------end of scale------#
test_x <- test_x %*% pca$rotation
test_x <- test_x[, 1:54]


#------evaluate OLS--------#
lm.model <- lm(SalePrice ~ ., data = as.data.frame(cbind(training_x,
    SalePrice = training_y)))
ols.predict <- predict(lm.model, newdata = as.data.frame(test_x))
mean((ols.predict - test_y)^2)
#------end evaluate OLS--------#
```

```r
#------evaluate ridge--------#
fit_ridge_cv <- cv.glmnet(training_x, training_y, alpha = 0)
best_ridge <- glmnet(training_x, training_y, alpha = 0,
    lambda = fit_ridge_cv$lambda.min)

best.lambda <- fit_ridge_cv$lambda.min
plot(fit_ridge_cv)
abline(v = log(best.lambda), col = "blue", lwd = 2)

ridge.pred <- predict(best_ridge, test_x)

mean((ridge.pred - test_y)^2)
#------end evaluate ridge--------#


#------evaluate lasso--------#
fit_lasso_cv <- cv.glmnet(training_x, training_y, alpha = 1)
best_lasso <- glmnet(training_x, training_y, alpha = 1,
    lambda = fit_lasso_cv$lambda.min)

best.lambda <- fit_lasso_cv$lambda.min
plot(fit_lasso_cv)
abline(v = log(best.lambda), col = "blue", lwd = 2)

lasso.pred <- predict(best_lasso, test_x)

mean((lasso.pred - test_y)^2)
#------end evaluate lasso--------#

#------evaluate elastic net--------#
fit_net_cv <- cv.glmnet(training_x, training_y, alpha = 0.5)
best_net <- glmnet(training_x, training_y, alpha = 1, lambda = fit_net_cv$lambda.min)

best.lambda <- fit_net_cv$lambda.min
plot(fit_net_cv)
abline(v = log(best.lambda), col = "blue", lwd = 2)

net.pred <- predict(best_net, test_x)

mean((net.pred - test_y)^2)
#------end elastic net--------#
```