MATH 242 Midterm Project

Jiaxing and Gage

3/02/2024

```r
# install.packages("leaps")
# install.packages("glmnet")
```

```r
library(dplyr)
library(readr)
library(ggplot2)
library(glmnet)
library(boot)
library(gridExtra)
library(leaps)
```

```r
nyc_condos <- read.csv("data/nyc-condos_s24.csv")
nyc_condos_full <- read.csv("data/full_data.csv")
```

```r
# Set the seed
set.seed(123)
# Create an index to randomly sample 70% of the data for training
train_index <- sample(1:nrow(nyc_condos_full), 0.7 * nrow(nyc_condos_full))
# Create the training set
train_data <- nyc_condos_full[train_index, ]
# Create the testing set
test_data <- nyc_condos_full[-train_index, ]
```

```r
# summary of dataset
str(nyc_condos)
```

```
## 'data.frame':    200 obs. of  16 variables:
##  $ Boro.Block.Lot         : chr  "1-01613-7501" "1-01171-7501" "3-02237-7519" "4-04955-7512" ...
##  $ Condo.Section          : chr  "0267-R1" "1058-R1" "3457-R1" "0278-R1" ...
##  $ Address                : chr  "1255 5 AVENUE" "200 RIVERSIDE BOULEVARD" "135 MIDDLETON STREET" "13
##  $ Neighborhood           : chr  "UPPER EAST SIDE (96-110)" "UPPER WEST SIDE (59-79)" "WILLIAMSBURG-C
##  $ Building.Classification: chr  "R4-CONDOMINIUM" "R4  -ELEVATOR" "R4-ELEVATOR" "R2-CONDOMINIUM" ...
##  $ Total.Units            : int  59 358 14 4 198 10 60 6 10 20 ...
##  $ Year.Built             : int  1925 1997 1942 1987 1963 1983 1928 1959 2005 2004 ...
##  $ Gross.SqFt             : int  63284 512280 26964 4010 206278 10962 61084 4497 9082 22295 ...
##  $ Estimated.Gross.Income : int  1613742 29871047 579187 60391 6266726 392220 742781 92683 242853 72
##  $ Gross.Income.per.SqFt  : num  25.5 58.3 21.5 15.1 30.4 ...
##  $ Estimated.Expense      : int  726500 5665817 205466 24782 2044215 162457 417204 37100 73837 21871
##  $ Expense.per.SqFt       : num  11.48 11.06 7.62 6.18 9.91 ...
##  $ Net.Operating.Income   : int  887242 24205230 373721 35609 4222511 229763 325577 55583 169016 505
```

```
##  $ Full.Market.Value      : int  6857996 196582995 2914000 239000 32481000 1826000 2048000 437001 135
##  $ Market.Value.per.SqFt  : num  108.4 383.7 108.1 59.6 157.5 ...
##  $ Report.Year            : int  2015 2019 2016 2012 2012 2015 2014 2018 2019 2012 ...
```

```
# # calculate average market value for each year
# nyc_condos <- nyc_condos %>%
#   group_by(Report.Year) %>%
#   mutate(average_market_value = mean(Full.Market.Value, na.rm = TRUE))
#
#
# # Log transform Gross SqFt
# nyc_condos$log_GrossSqFt <- log(nyc_condos$Gross.SqFt)
# # Log transform Estimated Gross Income
# nyc_condos$log_EstimatedGrossIncome <- log(nyc_condos$Estimated.Gross.Income)
# # Log transform Estimated Expense
# nyc_condos$log_EstimatedExpense <- log(nyc_condos$Estimated.Expense)
# Log transform Net Operating Income
nyc_condos$log_NetOperatingIncome <- log(nyc_condos$Net.Operating.Income)
nyc_condos_full$log_NetOperatingIncome <- log(nyc_condos_full$Net.Operating.Income)
train_data$log_NetOperatingIncome <- log(train_data$Net.Operating.Income)
test_data$log_NetOperatingIncome <- log(test_data$Net.Operating.Income)


nyc_condos$log_Full.Market.Value <- log(nyc_condos$Full.Market.Value)
nyc_condos_full$log_Full.Market.Value <- log(nyc_condos_full$Full.Market.Value)
train_data$log_Full.Market.Value <- log(train_data$Full.Market.Value)
test_data$log_Full.Market.Value <- log(test_data$Full.Market.Value)



nyc_condos$Net.Operating.Income.per.SqFt <- nyc_condos$Net.Operating.Income/ nyc_condos$Gross.SqFt
nyc_condos_full$Net.Operating.Income.per.SqFt <- nyc_condos_full$Net.Operating.Income/ nyc_condos_full$G
train_data$Net.Operating.Income.per.SqFt <- train_data$Net.Operating.Income/ train_data$Gross.SqFt
test_data$Net.Operating.Income.per.SqFt <- test_data$Net.Operating.Income/ test_data$Gross.SqFt



# Log transform Full Market Value
# nyc_condos$log_FullMarketValue <- log(nyc_condos$Full.Market.Value)
#
# nyc_condos$log_average_market_value <- log(nyc_condos$average_market_value)
```

```
set.seed(250)
## model.matrix() creates our design matrix of predictors
x <- model.matrix(Full.Market.Value  ~ Year.Built + Total.Units + Estimated.Gross.Income + Gross.Income
## select our outcome and convert it into a vector
## instead of a dataframe
y <- nyc_condos %>% select(Full.Market.Value) %>% unlist() %>% as.numeric()
## fit lasso for a range of lambda values (lambda is the tuning parameter
## that controls shrinkage)
cv.out <- cv.glmnet(x, y, alpha = 1)
## pick out the optimal lambda
bestlam <- cv.out$lambda.min
## get coefficients from the Lasso model
```

```r
lasso <- as.matrix(coef(cv.out, s = bestlam))
t(lasso)
```

```
##    (Intercept) Year.Built Total.Units Estimated.Gross.Income
## s1    -1836394    963.828   -5292.705               7.701503
##    Gross.Income.per.SqFt Estimated.Expense Expense.per.SqFt
## s1               10508.7         -6.664841                0
##    log_NetOperatingIncome Gross.SqFt
## s1              -43867.68 -0.8515914
```

```r
# lm <- lm(Full.Market.Value ~ Year.Built + Gross.SqFt+ Gross.Income.per.SqFt + Expense.per.SqFt + log_
# summary(lm)

lm <- lm(log_Full.Market.Value ~ Year.Built + Gross.SqFt+ Gross.Income.per.SqFt + Expense.per.SqFt + Ne
summary(lm)
```

```
##
## Call:
## lm(formula = log_Full.Market.Value ~ Year.Built + Gross.SqFt +
##     Gross.Income.per.SqFt + Expense.per.SqFt + Net.Operating.Income.per.SqFt,
##     data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.7992  -0.4673   0.0973   0.5956   3.3182
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   1.283e+01  1.348e-01  95.185  < 2e-16 ***
## Year.Built                    3.909e-04  6.739e-05   5.800 6.73e-09 ***
## Gross.SqFt                    4.829e-06  3.913e-08 123.421  < 2e-16 ***
## Gross.Income.per.SqFt        -2.024e-02  3.460e-02  -0.585   0.5585
## Expense.per.SqFt              5.109e-02  3.464e-02   1.475   0.1402
## Net.Operating.Income.per.SqFt  8.027e-02  3.460e-02   2.320   0.0204 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8094 on 22040 degrees of freedom
##   (56 observations deleted due to missingness)
## Multiple R-squared:  0.6192, Adjusted R-squared:  0.6191
## F-statistic:  7167 on 5 and 22040 DF,  p-value: < 2.2e-16
```

Our Model is:

$$

$\text{Market Value} = -103900000 + 8193 \times \text{Year.Built} - 4845 \times \text{Total.Units} + 107.4 \times \text{Gross.SqFt} + 553200 \times \text{Gross.Income.per.SqFt} - 51$

$$

```r
mse.lm.i <- function(i) {
  # Split the test data into k folds
  folds <- split(test_data, 1:nrow(test_data) %% 10)

  # Get the test data for the current fold
  fold_test_data <- folds[[i]]

  # Predict on the test data for the current fold
  yhat_test <- predict(lm, newdata = fold_test_data)

  # Compute the MSE for the current fold
  mse_fold <- mean((fold_test_data$Full.Market.Value - yhat_test)^2)

  return(mse_fold)
}

mse.lm.i(5)
```

```
## [1] NA
```

```r
Lev <- data.frame(hatvalues(lm)) %>%
ggplot(aes(x = 1:length(hatvalues(lm)), hatvalues(lm))) +
geom_point() +
labs(title = "Leverage",
x = "x", y = "Leverage")


stdresid <- data.frame(rstandard(lm)) %>%
ggplot(aes(x = 1:length(rstandard(lm)), rstandard(lm))) +
geom_point() +
labs(title = "Standardized Residuals",
x = "x", y = "Standardized Residuals")



studresid <- data.frame(rstudent(lm)) %>%
ggplot(aes(x = 1:length(rstudent(lm)), rstudent(lm))) +
geom_point() +
labs(title = "Studentized Residuals",
x = "x", y = "Studentized Residuals")

cooks <- data.frame(cooks.distance(lm)) %>%
ggplot(aes(x = 1:length(cooks.distance(lm)), cooks.distance(lm))) +
geom_point() +
labs(title = "Cooks Distance",
x = "x", y = "Cooks Distance")



p <- length(lm$coeff)
# n <- nrow(train_data)
n <- 22102
```

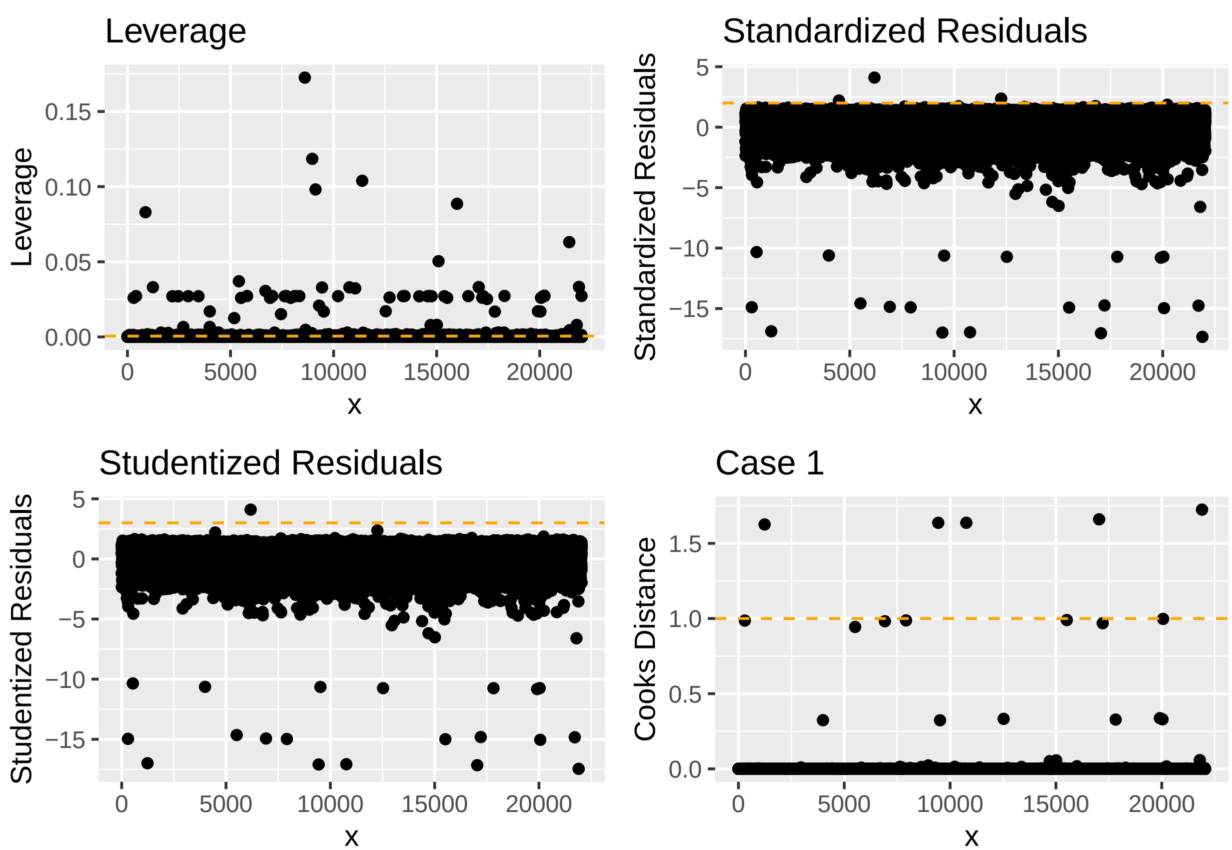
```r
Lev <- Lev +
geom_hline(yintercept = 2*p/n, lty = "dashed", col = "orange")

stdresid <- stdresid +
geom_hline(yintercept = 2, lty = "dashed", col = "orange")

studresid <- studresid +
geom_hline(yintercept = 3, lty = "dashed", col = "orange")

cooks <- cooks +
geom_hline(yintercept = 1, lty = "dashed", col = "orange") +
labs(title = "Case 1",
x = "x", y = "Cooks Distance")

grid.arrange(Lev, stdresid, studresid, cooks, ncol = 2)
```




```r
# high_leverage <- train_data %>% filter(hatvalues(lm) > 2* p/n)
# high_residuals <- train_data %>% filter(abs(rstandard(lm)) > 2)
# high_influence <- train_data %>% filter(cooks.distance(lm) > 1)
#
#
# print(high_leverage)
# print(high_residuals)
# print(high_influence)
```

Title:

Abstract

Introduction

Explanation of our variables:

1. **CondoSection**: Identification information for the condominium.
2. **Address**: Street address of the property.
3. **Neighborhood**: Name of the neighborhood where the property is located.
4. **BldgClassification**: Building classification code and description indicating the property's use.
5. **TotalUnits**: Total number of units in the building.
6. **YearBuilt**: Year the building was constructed.
7. **GrossSqFt**: Gross square footage of the building.
8. **EstGrossIncome**: Estimated gross income, calculated as income per square foot multiplied by gross square footage.
9. **GrossIncomePerSqFt**: Estimated gross income per square foot.
10. **EstimatedExpense**: Estimated expense, calculated as expense per square foot multiplied by gross square footage.
11. **ExpensePerSqFt**: Estimated expense per square foot.
12. **NetOperatingIncome**: Net operating income, calculated as estimated gross income minus estimated expense.
13. **FullMarketValue**: Current year's total market value of the property (land and building).
14. **MarketValuePerSqFt**: Market value per square foot, calculated as full market value divided by gross square footage.
15. **ReportYear**: Year of the report.
16. **Boro-Block-Lot**: Borough-Block-Lot location identifier for the property.

Methods

Results

Discussion

Conclusion

References