Multiple Linear Regression using R[*]

Jeric C. Briones

Department of Mathematics

Ateneo de Manila University

1 Preliminaries

1.1 Loading the Dataset

We begin by loading the dataset `BostonHousing` from the library `mlbench`. This dataset will be used for the regression analysis. Codes used are in `2Regression.R`.

The function `data()` loads the specified dataset and stores it to a variable with the same name as the dataset. In the example, the dataset `BostonHousing` is stored in the variable `BostonHousing`. The `names()` and `head()` functions provide an overview of the dataset.

```r
## loading the dataset
library(mlbench)
data(BostonHousing)
?BostonHousing #opens the R Documentation page of the dataset
names(BostonHousing) #column names of the dataset
head(BostonHousing) #prints first few rows the dataset
```

1.2 Exploratory Analysis and Preprocessing

After loading the dataset, we begin by looking at descriptive statistics of the dataset. This can be done using `summary()` and `plot()` functions.

```r
## exploratory analysis
summary(BostonHousing) #five-number summary for each column
plot(BostonHousing) #scatterplot for all possible variable pairs
```

After inspection, it can be seen that the variable `chas` is a binary variable. Since we are only interested with real-valued variables for our class discussions, the variable `chas` will be removed from the dataset. Putting a `-` before a column number (in this example, `4`) removes the specified column number.

```r
housing = BostonHousing[-4] #remove the binary variable 'chas'
```

We then proceed with checking the pairwise correlation of the variables. Recall that one of the assumptions of the multiple linear regression model is that the variables are uncorrelated. This can be done using the function `corrplot()` from the `corrplot` library.

[*]Supplementary notes for MATH 62.2 Time Series and Forecasting. Last updated Second Semester, SY 2023-2024.

```r
library(corrplot)
housing.corr = cor(housing) #correlation matrix
corrplot(housing.corr, method="color", type="upper") #plot correlation
```

Here, the parameter `type="upper"` is used to display the correlation matrix as a heatmap, while the parameter `type="upper"` is used to display only the upper triangle portion of the correlation matrix heatmap.

2 Fitting the Linear Model

Linear regression is primarily done using the `lm()` function. Regression results are then retrieved using the `summary()` function. To fit a linear model, the linear regression model must first be specified. For example, we are given the data matrix $\boldsymbol{X}$, where $\boldsymbol{X}$ is $n \times (p+1)$. If we are interested to use only three of the variables available and fit the model $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$, we will specify the model as `y ~ x1 + x2 + x3`. Note that the function includes an intercept term by default.

On the other hand, if we are interested to fit a full model (that is, use all of the variables available), the model will be specified as `y ~.`, where the `.` indicates that all other variables except for y will be used as independent variables for the regression.

2.1 Full Linear Model

Suppose we are interested with having `medv` as our dependent variable. This model will be specified as `medv~.`. This will then be the first parameter in the `lm()` function.

```r
## initial model
housing.full = lm(medv~., data=housing)
housing.full.res = summary(housing.full)
names(housing.full)  #available results obtained from regression
names(housing.full.res)  #available results obtained from regression
```

Here, the parameter `data=housing` is used to specify that the observations are stored in the variable `housing`. If the variables used in the model specification already exist in the current workspace, the parameter `data` can be omitted.

The predicted value vector $\hat{\boldsymbol{y}}$ is given by `housing.full$fitted.values`, the error vector $\boldsymbol{\varepsilon}$ is either `housing.full$residuals` or `housing.full.res$residuals`, while the the coefficient vector $\boldsymbol{\beta}$ is `housing.full$coefficients`. On the other hand, the F statistic for the hypothesis test and the associated degrees of freedom can be seen in `housing.full.res$fstatistic`, while the individual t-statistic for each of the coefficients are in `housing.full.res$coefficients`. Finally the adjusted R^2 and R^2 are in `housing.full.res$adj.r.squared` and `housing.full.res$r.squared`, respectively

```r
## regression results
housing.full$coefficients #regression coefficients
housing.full.res$fstatistic #F-test for linear hypothesis
housing.full.res$coefficients #generalized linear hypotheses
housing.full.res$adj.r.squared # adjR2
```

Inspecting `housing.full.res$coefficients`, it can be observed that the coefficients of `indus` and `age` are insignificant since their p-values (0.513889 and 0.834407, respectively) are greater than $\alpha = 0.05$. In such cases, it is possible to remove these variables and use the reduced model instead.

2.2 Reduced Linear Model

Suppose we are now interested with having medv as our dependent variable, and all other variables except for age as our independent variables. This new model will be specified as medv~.-age. The . still indicates that all other variables except for medv will be used as independent variables. However, by adding -age, the variable age is subtracted from the model. Similarly, if we also want to exclude indus from the model, this second new model will be specified as medv~.-age-indus.

```
## reduced models
housing.m11 = lm(medv~.-age, data=housing) #model w/o age
housing.m10 = lm(medv~.-age-indus, data=housing) #model w/o age, indus
housing.m11.res = summary(housing.m11)
housing.m10.res = summary(housing.m10)
housing.m11.res
housing.m10.res
```

Inspecting the reduced models housing.m11.res and housing.m10.res and comparing them against the full model housing.full.res, it can be seen that the reduced models have higher adjusted R^2 (0.7296032 and 0.7299149 against 0.7290788, respectively). Moreover, notice that in housing.m11.res, the coefficient of indus is still not significant.

To formally compare the fit of the two models (full vs reduced), the function anova() can be used. Note that the order of the model inputs does not affect the results. Moreover, it also assumed that one model is *contained* in the other. That is, all the variables in the smaller model is in the larger model. In this test, the null hypothesis H_0 is that the two models have the same predictive power. Thus, if the resulting p-value is small, we reject H_0 and say that one model is better than the other.

```
## comparing the models
housing.full.res$adj.r.squared
housing.m11.res$adj.r.squared
housing.m10.res$adj.r.squared
anova(housing.full, housing.m11) #full vs reduced model (no age)
anova(housing.full, housing.m10) #full vs reduced model (no age, indus)
```

Inspecting the results of the comparison for housing.full and housing.m11, it can be seen that we failed to reject H_0 (since p-value $= 0.7898$ is large). Thus, the full model and reduced model (the model with age and indus excluded) have the same predictive power. Thus, by *principle of parsimony*, we could then choose the reduced model.

An alternative interpretation for this result is as follows: since the reduced model and the full model have the same predictive power, the variables added to the reduced model does not increase its predictive power. Thus, we can just ignore those variables (and by extension, the full model) and use the reduced model instead. As such, we then write the functional form of the reduced model housing.m10 as

$$\begin{aligned} y = 36.62031 - 0.11406x_{\text{crim}} &+ 0.04574x_{\text{zn}} - 16.46915x_{\text{nox}} \\ &+ 3.84464x_{\text{rm}} - 1.5261x_{\text{dis}} + 0.31553x_{\text{rad}} - 0.01267x_{\text{tax}} \\ &- 0.97844x_{\text{ptratio}} + 0.00973x_{\text{b}} - 0.5281x_{\text{lstat}} + \varepsilon. \end{aligned} \tag{1}$$

In some cases, this can also be written as $\hat{y} = 36.62031 - 0.11406x_{\text{crim}} + \cdots - 0.5281x_{\text{lstat}}$, with $\hat{y}$ being used to indicate that the equation is for the *fitted* model.

3 Model Diagnostics

3.1 Variable Selection

Another way to select which variables to include in the regression is by using the function `regsubsets()` from the library `leaps`. The full model is specified as the first parameter for this function.

```r
## variable selection
library(leaps)
nvar = ncol(housing) - 1
varsel = regsubsets(medv~., data=housing, nvmax=nvar)
varsel.res = summary(varsel)
names(varsel.res) #available results from selection
```

Here, the full model `medv~.` is used as the first parameter. Moreover, the parameter `data=housing` is used to specify that the observations are stored in the variable `housing`. Lastly, the parameter `nvmax=nvar` is used to specify that the maximum number of variables to consider is `nvar`, which is one less the number of variables in `housing`. Note that the intercept term is not included in the variable count.

In the variable selection process, we consider adjusted R^2, Mallow's C_p, and BIC as our metric for choosing. Recall that the *best* model is the model with the highest adjusted R^2, the lowest Mallow's C_p that is closest to p, or the lowest BIC. Results for each number of variables included in each metric are stored in `$adjr2`, `$cp`, and `$bic`, respectively. For example, `varsel.res$bic[2]` returns the BIC when two variables are included in the model. On the other hand, the selected variables for a given number of variables included can be seen in `$which`. For example, `varsel.res$which[5,]` lists down the five variables included in the model.

```r
# compile results
varsel.metric = cbind(1:nvar, varsel.res$adjr2, varsel.res$cp,
                      varsel.res$bic)
colnames(varsel.metric) = c("No. of Variables", "Adjusted R2",
                            "Mallow's Cp", "BIC")
varsel.metric

# plot of each metric used
plot(varsel.res$adjr2) #adjR2
plot(varsel.res$cp) #Mallow's Cp
plot(varsel.res$bic) #BIC

# list of suggested variables for the results of each metric used
varsel.res$which[which.max(varsel.res$adjr2),] #adjR2
varsel.res$which[which.min(varsel.res$bic),] #BIC
```

Here, the functions `which.max()` and `which.min()` are used to find $\arg\max$ for adjusted R^2 and $\arg\min$ for BIC. Inspecting the results, `varsel.res` suggests that, based on the three aforementioned metrics, ten variables should be used as independent variables. Moreover, the variables included in the results are similar to our earlier observations. That is, the variables `age` and `indus` are suggested to be excluded from the model.

3.2 Analyzing Residuals

Aside from variable selection, we are also concerned with the distribution of the residuals. Recall that one of the assumptions of multiple linear regression model is that the residuals are normally distributed and are uncorrelated. That is, if the residuals are uncorrelated, there should be no evident patterns in their plot. To check these assumptions, we can use the functions `qqnorm()`, and `ad.test()` and `shapiro.test()` from the library `nortest`.

The `qqnorm()` is used for to generate the normal *Q-Q* plot. If the residuals follow a normal distribution, its quantiles should match the theoretical quantiles (as indicated by `qqline`) based from the normal distribution. On the other hand, the Anderson-Darling and Shapiro-Wilk are statistical tests with the null hypothesis that the data comes from a normal distribution.

```
## analyzing residuals
library(nortest)
plot(housing.m10$residuals) #check if there are patterns
qqnorm(housing.m10$residuals) #Q-Q plot of data
qqline(housing.m10$residuals) #Q-Q line
ad.test(housing.m10$residuals) #Anderson-Darling
shapiro.test(housing.m10$residuals) #Shapiro-Wilk
```

Inspecting the results, it seems like the residuals do not follow the normal distribution. This is supported by the small p-values of the two tests, as well as having the tails of the the *Q-Q* plot deviate away from the normal line.

3.3 Testing Multicollinearity

Lastly, we test for multicollinearity using the function `vif` from the library `car`. The VIF of a predictor is a measure for how easily it is predicted from a linear regression using the other predictors. Taking the square root of the VIF tells you how much larger the standard error of the estimated coefficient is with respect to the case when that predictor is independent of the other predictors.

A general guideline is that a VIF ≥ 5 or VIF ≥ 10 is large, indicating that the model has problems estimating the coefficient. However, this in general does not degrade the quality of predictions. If the VIF is larger than $\dfrac{1}{1-R^2}$, where R^2 is the multiple R^2 of the regression, then that predictor is more related to the other predictors than it is to the response.

```
## multicollinearity
library(car)
vif(housing.m10)

# if variables w/ vif>5 are excluded
housing.m8 = lm(medv~.-age-indus-rad-tax,data=housing)
vif(housing.m8) #vif of the new model
summary(housing.m8)$adj.r.squared #adjR2 of new model
```

Inspecting the results, it can be seen that the VIF for `rad` and `tax` are both greater than 5. If these variables are excluded, the resulting model (as seen in `housing.m8`) have lower VIF's, at the expense of having a lower adjusted R^2.