Unlike the training RMSE, the test RMSE first decreases from Models 1 to 4, then eventually rises when going from Models 4 to 5. As we make our linear model more sophisticated, its predictive performance initially improves due to its increased ability to capture complicated patterns. When it becomes unnecessarily complicated, it appears to overfit the data and its predictive performance worsens. This creates the characteristic U-shape that characterizes the test error as a function of model complexity discussed in Subsection 3.1.3. Among the five linear models, Model 4 gives rise to the *smallest* test RMSE and is our recommended linear model on the basis of prediction accuracy. Granted, there may be (and very likely there are!) other specifications of `TV` and `radio` that result in an even more predictive model for `sales`, but Model 4 is arguably the best linear model in terms of prediction accuracy among the five models above.

3.4 Case Study 2: Feature Selection and Regularization

Having looked at a warm-up case study of linear models, we now turn to a substantially more extended case study whose sophistication closely resembles that of a typical PA exam project. In addition to the modeling concepts discussed in the advertising case study in Section 3.3, this section will also illustrate how to deal with categorical predictors in a linear model, how to select useful features from a large set of predictors, and how to perform regularization, all of which are popular test items in Exam PA. Specifically, after completing this case study, you should be able to:

- Fit a multiple linear regression model with both quantitative and qualitative predictors.
- Detect and accommodate interactions between predictors which can be quantitative or qualitative.
- Binarize categorical predictors using the `dummyVars()` function from the `caret` package and recognize the need for explicit binarization.
- Select useful features using the `stepAIC()` function from the `MASS` package and be familiar with the different options allowed by this function.
- Generate and interpret diagnostic plots for a linear model.
- Implement ridge regression and lasso using the `glmnet()` and `cv.glmnet()` functions from the `MASS` package.

To mimic the exam environment and make our learning more systematic, this case study (and the case studies in subsequent chapters) will be organized around a series of well-defined tasks, which are displayed in boxes and worded in a way that parallels released PA exams.

3.4.1 Preparatory Work

Data description. In this section, we will examine the `Credit` dataset that accompanies Chapters 3 and 6 of ISLR. Compared to the advertising dataset in Section 3.3, the `Credit` dataset has a lot more predictors, some of which are categorical, and is a perfect illustration of feature selection. Again, you may have already seen the `Credit` data when studying for Exam SRM, but our treatment here will have a much heavier predictive analytic flavor and will be structured in a way that is tailored for Exam PA (e.g., we will only make use of R functions covered by the PA e-learning modules and exam projects, which are different from the R functions used in ISLR).

First of all, run CHUNK 1 to read in the dataset[xxiv] and print a summary of each variable.

```
# CHUNK 1
Credit <- read.csv("Credit.csv")
# OR load the data directly from the ISLR package and delete the first column
#library(ISLR)
#data(Credit)

summary(Credit)

##      Income           Limit           Rating          Cards
##  Min.   : 10.35   Min.   :  855   Min.   : 93.0   Min.   :1.000
##  1st Qu.: 21.01   1st Qu.: 3088   1st Qu.:247.2   1st Qu.:2.000
##  Median : 33.12   Median : 4622   Median :344.0   Median :3.000
##  Mean   : 45.22   Mean   : 4736   Mean   :354.9   Mean   :2.958
##  3rd Qu.: 57.47   3rd Qu.: 5873   3rd Qu.:437.2   3rd Qu.:4.000
##  Max.   :186.63   Max.   :13913   Max.   :982.0   Max.   :9.000
##       Age          Education        Gender    Student   Married
##  Min.   :23.00   Min.   : 5.00   Female:207   No :360   No :155
##  1st Qu.:41.75   1st Qu.:11.00   Male  :193   Yes: 40   Yes:245
##  Median :56.00   Median :14.00
##  Mean   :55.67   Mean   :13.45
##  3rd Qu.:70.00   3rd Qu.:16.00
##  Max.   :98.00   Max.   :20.00
##                Ethnicity      Balance
##  African American: 99   Min.   :   0.00
##  Asian           :102   1st Qu.:  68.75
##  Caucasian       :199   Median : 459.50
##                         Mean   : 520.01
##                         3rd Qu.: 863.00
##                         Max.   :1999.00
```

The `Credit` dataset contains $n = 400$ observations and 11 variables described in the data dictionary shown in Table 3.4. The target variable is the last variable, `Balance`, a quantitative, integer-valued variable that ranges from 0 to 1,999 and represents the credit card debt for an individual. The other variables can be grouped as follows:

1. The first six variables, `Income`, `Limit`, `Rating`, `Cards`, `Age`, and `Education`, are quantitative predictors for `Balance`.

2. The other four variables, `Gender`, `Student`, `Married`, and `Ethnicity`, are qualitative predictors with either two or three levels.

Our goal here is to identify and interpret *key* financial and demographic factors that relate to a higher or lower `Balance` with the aid of appropriate linear models.

In some situations, `Ethnicity` may be a sensitive predictor to use—making predictions on the basis of ethnicity may be criticized on the grounds of discrimination! For this case study, we will

[xxiv] The CSV file was downloaded from `http://faculty.marshall.usc.edu/gareth-james/ISL/Credit.csv` and its first column (simply a column of row indices) has been deleted.

Variable	Description	Characteristics
`Income`	Income ($10,000's)	A positive number with two decimal places
`Limit`	Credit limit	A positive integer from 855 to 13,913
`Rating`	Credit rating	A positive integer from 93 to 982
`Cards`	Number of credit cards	A positive integer from 1 to 9
`Age`	Age in years	A positive integer from 23 to 98
`Education`	Number of years of education	A positive integer from 5 to 20
`Gender`	An indicator of the individual's gender	A factor with levels `Male` and `Female`
`Student`	An indicator of whether the individual was a student	A factor with levels `No` and `Yes`
`Married`	An indicator of whether the individual was married	A factor with levels `No` and `Yes`
`Ethnicity`	An indicator of the individual's ethnicity	A factor with levels `African American`, `Asian`, and `Caucasian`
`Balance`	Average credit card balance ($)	A positive integer from 0 to 1999

Table 3.4: Data dictionary for the `Credit` data.

take `Ethnicity` as an ordinary predictor and see if it may prove useful for predicting `Balance`. (Spoiler alert: It is not useful!)

TASK 1: Explore the relationship of each predictor to `Balance`

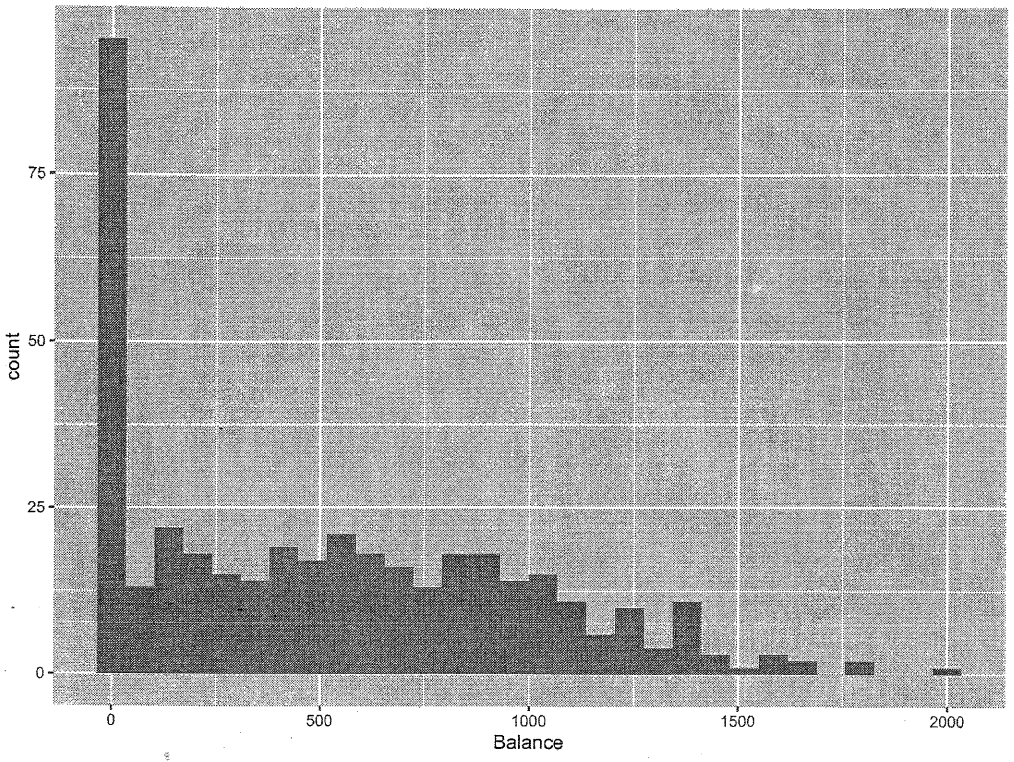
Use graphical displays and summary statistics to form preliminary conclusions regarding which variables are likely to have significant predictive power.

Data exploration: Distribution of the target variable. This preparatory task is a version of Task 1 of the June 2019 PA exam. To begin with, let's explore the distributional characteristics of the target variable `Balance`. Note that although not directly requested in the question statement, the exploration of the distribution of the target variable is expected by the SOA as the nature of the target variable may suggest useful transformations and determine which predictive analytic techniques should be applied (the June 2019 exam model solution says, *"[m]any candidates failed to consider the skewed distribution of the target variable before launching into the analysis."*). Needless to say, the target variable is by far the most important variable in the data!

From the summary in CHUNK 1, the median, mean, and maximum of `Balance` are 459.50, 520.01, and 1,999, respectively. The fact that the mean exceeds the median suggests that the distribution of `Balance` is skewed to the right. This is confirmed by the histogram of `Balance` constructed in CHUNK 2 (see Figure 3.4.1). However, there are a large number of observations (90 to be exact) for which `Balance` equals zero. This makes the log transformation problematic and so we will leave `Balance` as is.

Data exploration: Target variable and numeric predictors. Now let's turn to the relationship between each predictor and the target variable, which involves bivariate data exploration, with the aid of graphs and summary statistics. As we learned in Section 2.2, the best way to explore the

```
# CHUNK 2
library(ggplot2)
ggplot(Credit, aes(x = Balance)) +
  geom_histogram()
```




```
nrow(Credit[Credit$Balance == 0, ])

## [1] 90
```

Figure 3.4.1: A histogram for the target variable `Balance` in the `Credit` dataset.

relationship between two numeric variables is a scatterplot. In CHUNK 3, we use a `for` loop (recall Section 1.4) to make a scatterplot, with a smoothed line superimposed, for `Balance` against each of the six numeric predictors (see Figure 3.4.2). Similar code was provided in the June 2019 PA exam. We can see that `Balance` has a strong positive linear relationship with both `Limit` and `Rating`. There is also a positive linear relationship between `Balance` and `Income`, but its strength is not as strong and there are hardly any systematic patterns between `Balance` and each of `Cards`, `Age`, and `Education`. At this stage, it appears that `Income`, `Limit`, and `Rating` are important predictors affecting `Balance` favorably.

In fact, the positive linear relationships for `Limit` and `Rating` look so similar that the two predictors seem to be almost carbon copies of each other. This can be checked numerically by computing their correlation or graphically by plotting the two variables in comparison with each other, as we do in CHUNK 4. The scatterplot in Figure 3.4.3 reveals that the two variables are virtually perfectly related in a linear fashion. Intuitively, using two almost perfectly linearly related

```
# CHUNK 3
# first save the names of the numeric predictors as a vector
vars.numeric <- colnames(Credit[, 1:6])
for (i in vars.numeric) {
  plot <- ggplot(Credit, aes(x = Credit[, i], y = Balance)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE) +
    labs(x = i)
  print(plot)
}
```

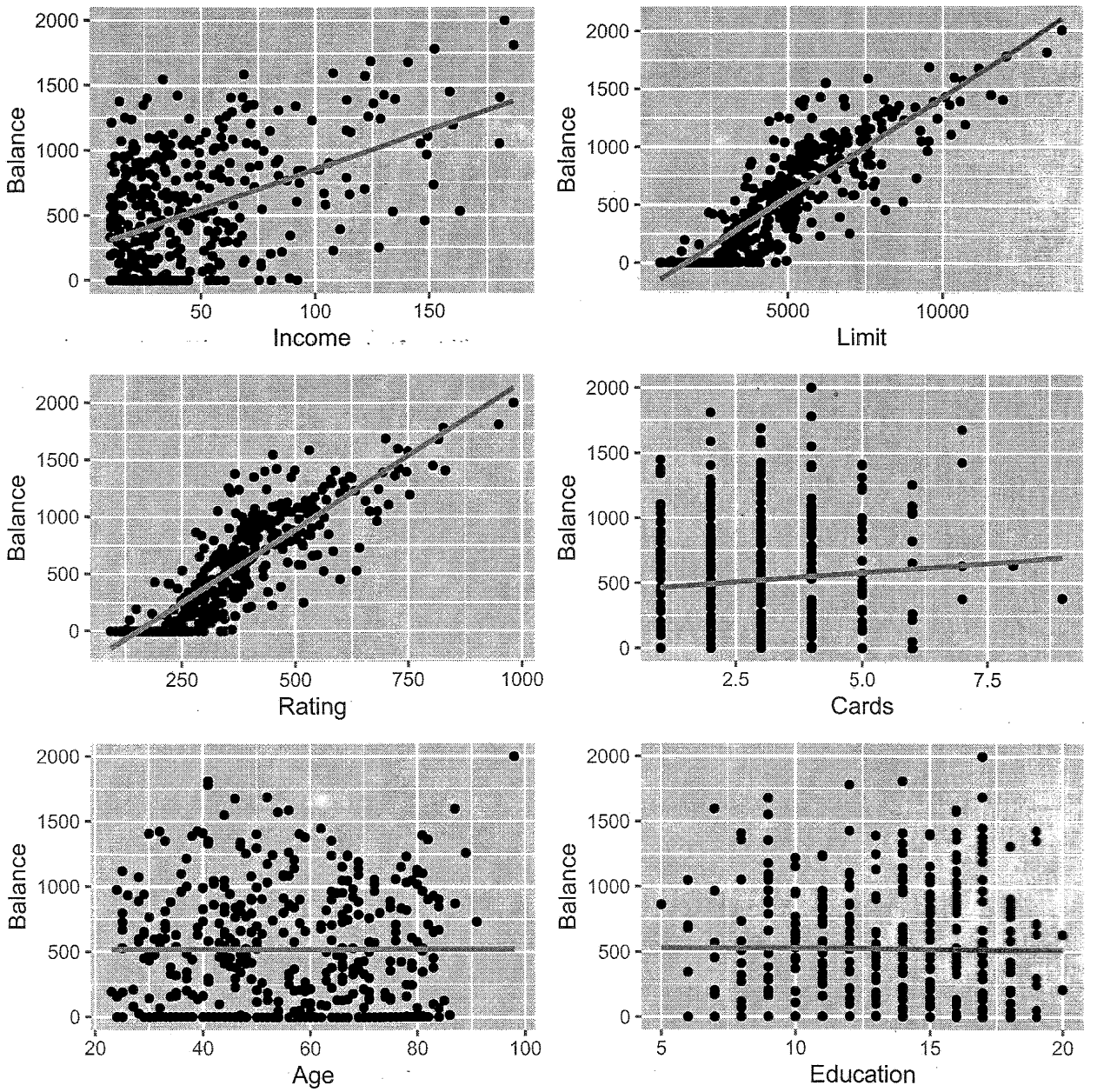


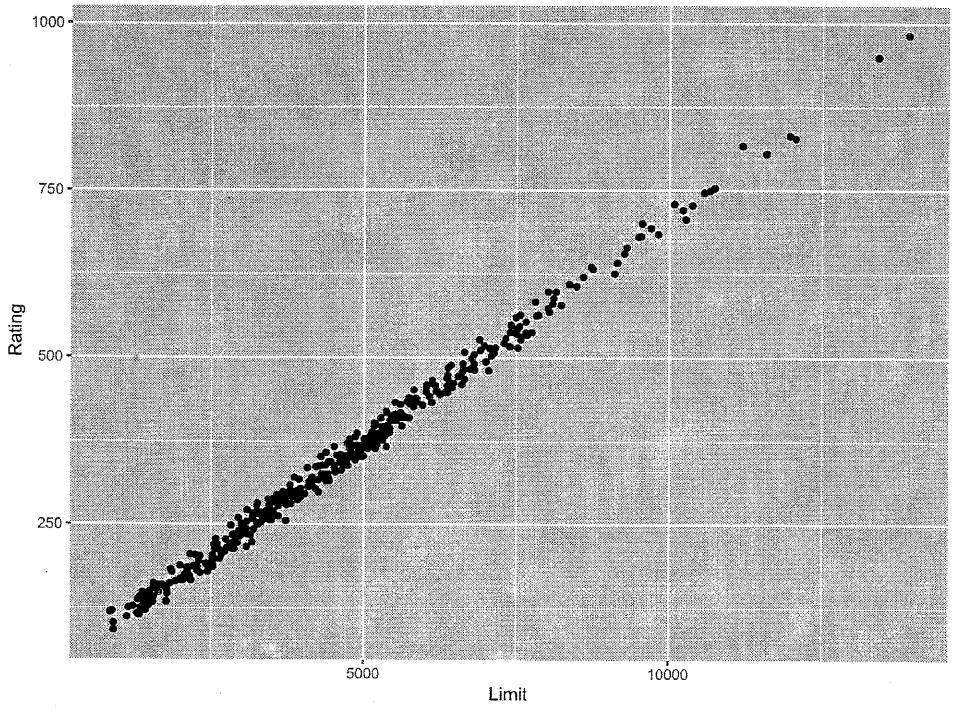
Figure 3.4.2: Scatterplots between `Balance` and the six numeric predictors (`Income`, `Limit`, `Rating`, `Cards`, `Age`, and `Education`) in the `Credit` dataset.

predictors does not bring much additional predictive power compared to using only one of them. Technically, the co-existence of two such predictors in a linear model can raise the issue of *collinearity* (a topic in Exam SRM not discussed in detail in the PA e-learning modules), which destabilizes the model fitting procedure and may lead to counter-intuitive model results. Therefore, we will keep one and only one of the two predictors, say `Rating`, and delete `Limit`, as we do in the last line of CHUNK 4. (If you are curious, you may delete `Rating` and retain `Limit` and repeat all of the analysis in the rest of this section.)

```
# CHUNK 4
cor(Credit$Limit, Credit$Rating)

## [1] 0.9968797

ggplot(Credit, aes(x = Limit, y = Rating)) +
  geom_point()
```




```
Credit$Limit <- NULL
```

Figure 3.4.3: A scatterplot for `Limit` and `Rating` in the `Credit` dataset.

Data exploration: Target variable and categorical predictors. To explore the relationship between `Balance` and the four categorical predictors, we will make use of split box plots, where a series of box plots for `Balance` indexed by the levels of each categorical predictor are produced, as we do in CHUNK 5 (see Figure 3.4.4). Of the four categorical predictors, only `Student` stands out as making a significant difference to `Balance` over the two levels, `Yes` and `No`, with the average `Balance` being much higher for students (`Student = Yes`) than for non-students (`Student = No`).

```
# CHUNK 5
# Save the names of the categorical predictors as a vector
vars.categorical <- c("Gender", "Student", "Married", "Ethnicity")
for (i in vars.categorical) {
  plot <- ggplot(Credit, aes(x = Credit[, i], y = Balance)) +
    geom_boxplot() +
    labs(x = i)
  print(plot)
}
```

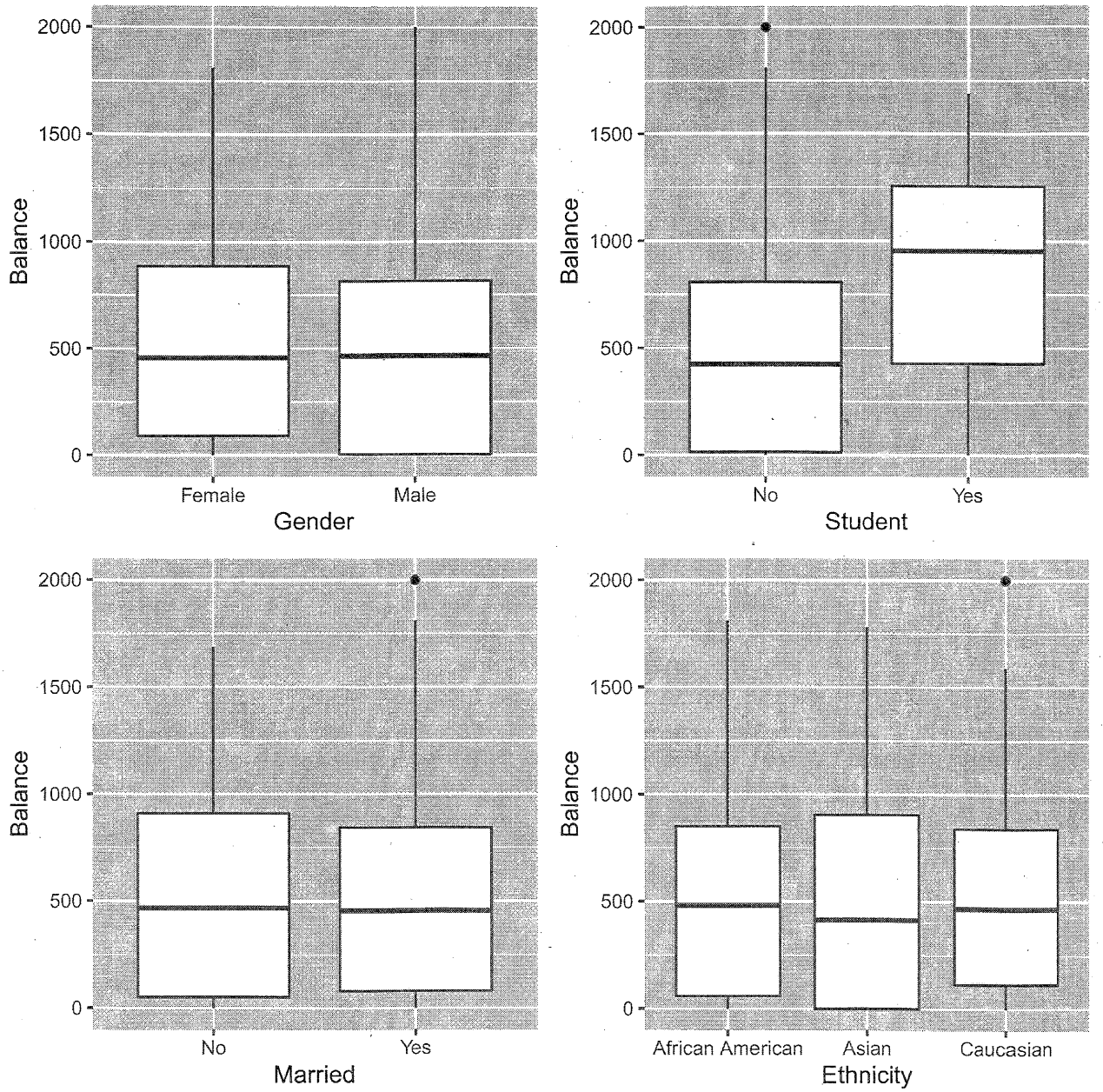


Figure 3.4.4: Box plots for `Balance` split by the levels of the four categorical predictors (`Gender`, `Student`, `Married`, and `Ethnicity`) in the `Credit` dataset.

Overall, the most promising predictors for `Balance` are `Income`, `Rating`, and `Student`.

TASK 2: Select an interaction

Select one or more pairs of features that may be included as an interaction variable in a multiple linear regression model. Do this by first proposing two variables that are likely to interact and then using appropriate graphical displays to confirm the existence of an interaction. Continue until one or more promising interactions have been identified. Include your selected interaction variable(s) when constructing a linear model in subsequent tasks.

Recall that an interaction means that the expected effect of one predictor on the target variable depends on the value or level of another predictor *(Note: It is a good idea to begin your response by stating the definition of interaction, as in SOA's June 2019 model solution. This not only guides your investigation, but also provides a good opportunity for you to demonstrate to the SOA your knowledge of predictive analytics!).* To look for promising interactions, let's focus on the three important predictors identified in Task 1, `Income`, `Rating`, and `Student`. It may be that students, with a different financial status from non-students, form a group where the sensitivity of `Balance` to `Income` and `Rating` differs vastly *(Note: It is a good idea to form conjectures driven by practical considerations.).* As noted earlier, it is possible that significant interactions exist for predictors whose main effects are insignificant. It will, however, make your life much easier if you start with those significant predictors. As soon as an obvious interaction has been found, you can stop, do your write-up, and proceed to the next task.

To investigate whether `Student` (categorical) interacts with `Income` (numeric) and `Rating` (numeric), in CHUNK 6 we make two scatterplots, one for `Balance` against `Income` and one for `Balance` against `Rating`, and color-distinguish the data points on the basis of the two levels of `Student`, as in Figure 3.4.5. In each plot, a smoothed line is superimposed for every level of `Student` to ease inspection. In the first scatterplot, the slopes of the two smoothed lines differ quite remarkably, with the non-students being more sensitive to `Income` than students. This suggests including the interaction variable `Income:Student`. The interaction between `Rating` and `Student` is less pronounced, but we may still include the interaction variable `Rating:Student` and let the linear model decide whether or not to retain this feature at a later stage.

To conclude, we will add two interaction variables, `Income:Student` and `Rating:Student`, to the linear models we are going to construct. To be sure, there may be other promising interaction variables (or polynomial terms!) that can be added to the model to improve its predictive performance. Given the 5-hour time frame available on the exam, the task you are asked to complete will probably require that you identify one (or two) pair of interacting predictors. There is no need to try out all combinations.

```
# CHUNK 6
p1 <- ggplot(Credit, aes(x = Income, y = Balance, color = Student)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)

p2 <- ggplot(Credit, aes(x = Rating, y = Balance, color = Student)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)

library(gridExtra)
grid.arrange(p1, p2, ncol = 2)
```

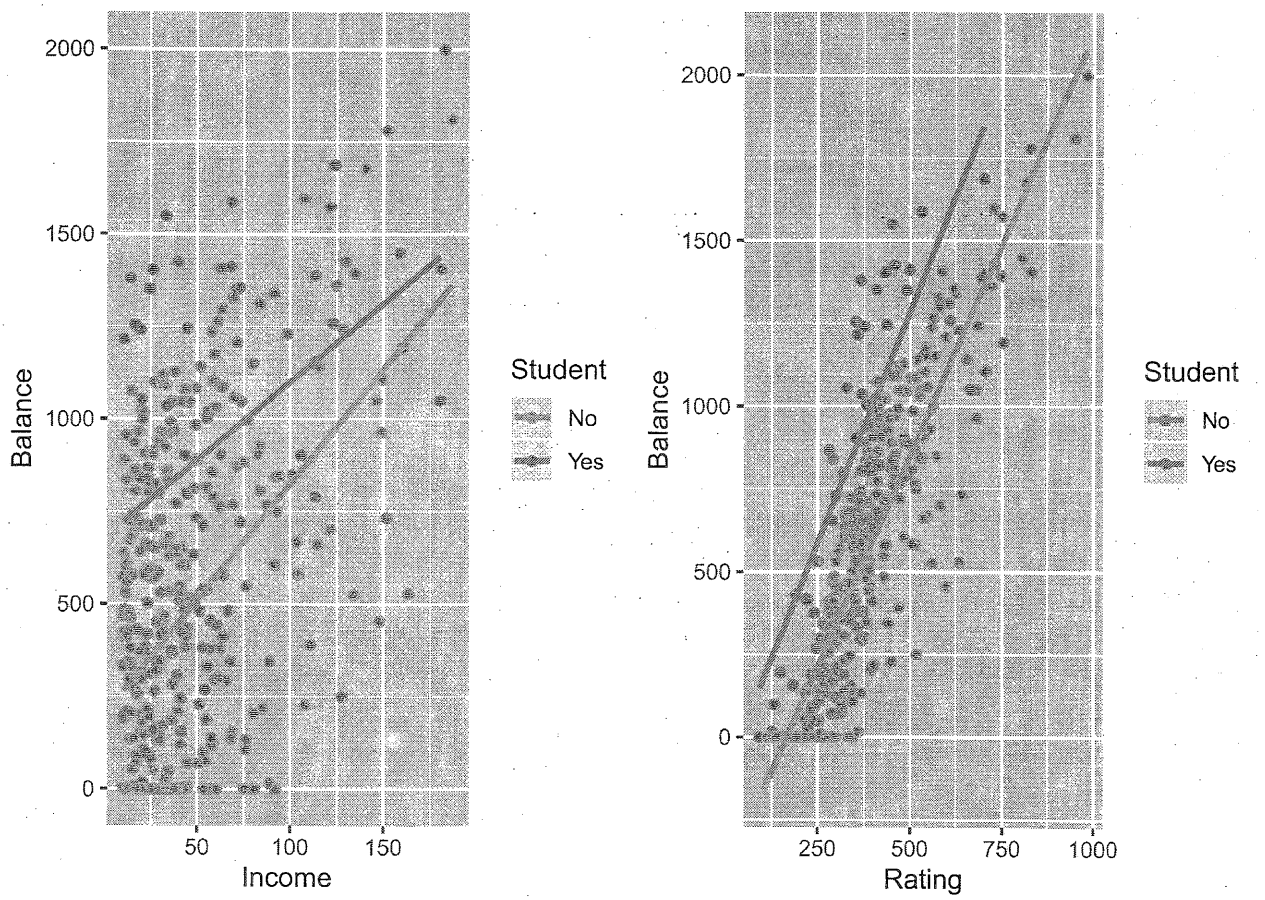


Figure 3.4.5: Scatterplots of `Balance` against `Income` (left) and `Rating` (right) in the `Credit` dataset.

Creation of training and test sets. To pave the way for evaluating the predictive performance of different linear models that will come later, we conclude this preparatory subsection by setting up the training and test sets, again by the `createDataPartition()` function first seen in Subsection 3.3.3. This is done in CHUNK 7.

```
# CHUNK 7
library(caret)
set.seed(42)
partition <- createDataPartition(Credit$Balance, p = 0.75, list = FALSE)
train <- Credit[partition, ]
test <- Credit[-partition, ]

print("TRAIN")
mean(train$Balance)

print("TEST")
mean(test$Balance)

## [1] "TRAIN"
## [1] 531.1362
## [1] "TEST"
## [1] 486.202
```

Again, the mean of `Balance` in the training set is pretty close to that in the test set due to the built-in stratification, though not as close as we may have expected. This can be attributed to the rather skewed nature of `Balance`.

3.4.2 Feature Selection

Fitting the first model. In CHUNK 8, we fit the linear model with all of the $p = 11$ eleven features (9 variables in the original dataset and 2 interaction variables), save the fitted model as `model.full` (meaning the "full model"), and output a model summary. Note that when specifying the two interaction terms in the formula, it is enough to use the colon operator instead of the asterisk because the placeholder "." already includes the main effects terms. (In fact, if you change `Income:Student + Rating:Student` to `Income*Student + Rating*Student`, the model summary will remain the same. Try it!)

```
# CHUNK 8
model.full <- lm(Balance ~ . + Income:Student + Rating:Student, data = train)
summary(model.full)

##
## Call:
## lm(formula = Balance ~ . + Income:Student + Rating:Student, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -204.83  -67.69  -10.17   60.36  300.13
```

```
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -568.87494   39.26643 -14.488  < 2e-16 ***
## Income              -7.41624    0.27847 -26.633  < 2e-16 ***
## Rating               3.93051    0.06245  62.936  < 2e-16 ***
## Cards                5.52797    4.12314   1.341  0.18107
## Age                 -0.73763    0.33635  -2.193  0.02910 *
## Education           -0.30429    1.81060  -0.168  0.86665
## GenderMale          10.56953   11.31508   0.934  0.35103
## StudentYes         235.25194   53.25361   4.418 1.41e-05 ***
## MarriedYes          -4.01562   11.86499  -0.338  0.73528
## EthnicityAsian      14.61313   16.33913   0.894  0.37187
## EthnicityCaucasian  19.08945   14.06755   1.357  0.17585
## Income:StudentYes   -2.47014    0.80176  -3.081  0.00226 **
## Rating:StudentYes    0.85412    0.20479   4.171 4.02e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 97.26 on 288 degrees of freedom
## Multiple R-squared:  0.9601,Adjusted R-squared:  0.9584
## F-statistic: 577.2 on 12 and 288 DF,  p-value: < 2.2e-16
```

Note that the `lm()` function automatically turns categorical predictors such as `Gender`, `Student`, `Married`, and `Ethnicity` into dummy variables, which then enter the equation of the multiple linear regression model. These dummy variables are named in the form of `<cat_predictor><level>`. For example, `StudentYes` is a dummy variable that equals 1 if the observation in question is a student and 0 otherwise. By default, a categorical predictor with c levels is represented by $c-1$ dummy variables, each of which is an indicator of one and only one of the $c-1$ levels of the predictor. The level that is left out is the baseline level, which, by default, is the alpha-numerically first level (or the first level in the `levels` argument if the factor is created by the `factor()` function; see page 19). For instance, the baseline level of `Student` is `No`, which comes before `Yes` in alphabetical order; this explains why instead of `StudentNo`, the `StudentYes` variable appears in the model output. That the coefficient estimate for `StudentYes` is 235.25194 means that the credit card balance of students are higher than that of non-students (baseline) by 235.25194 on average, *holding all other predictors fixed*. In the same way, the baseline levels of `Gender`, `Married`, and `Ethnicity` are `Female`, `No`, and `African American`, respectively. (If you go back to the frequency tables of the four categorical predictors in CHUNK 1, you can see that the baseline levels are listed first.)

The model summary also agrees with the findings in the data exploration task we completed (Task 1) and shows that `Income`, `Rating`, `Student` (or equivalently, `StudentYes`), and the two interaction terms are highly significant, meaning that they may be valuable features that can be included to improve prediction performance.

Releveling. In all of the PA sample and exam projects released by the SOA, a code chunk is supplied for you to *relevel* factor variables so that their baseline level is not the alpha-numerically first level, but the *level with the most observations*. While releveling has no effect on the predicted values, changing the baseline levels will affect the coefficient estimates of the dummy variables

(which are now dummy variables with respect to another baseline level) and, more importantly, the significance statistics such as t-statistics for the dummy variables and the corresponding p-values. Remember, the p-values measure how different the non-baseline levels are from the baseline level, and changing the baseline level certainly changes the measurement of such differences. In general, for linear and GLMs the best practice is to set the baseline level to the most populous level to improve the accuracy of our measures of significance.

In CHUNK 9, we apply SOA's code chunk to the `Credit` data and relevel the four categorical predictors using a `for` loop and the `relevel()` function.

```
# CHUNK 9
for (i in vars.categorical){
  # Use the table() function to calculate the frequencies for each factor
  table <- as.data.frame(table(Credit[, i]))
  # Determine the level with the highest frequency
  max <- which.max(table[, 2])
  # Save the name of the level with the highest frequency
  level.name <- as.character(table[max, 1])
  # Set the baseline level to the most populous level
  Credit[, i] <- relevel(Credit[, i], ref = level.name)
}

summary(Credit[, vars.categorical])
```

```
##     Gender    Student   Married              Ethnicity
##  Female:207   No :360   Yes:245   Caucasian       :199
##  Male  :193   Yes: 40   No :155   African American: 99
##                                   Asian           :102
```

As you can see, the baseline levels of `Married` and `Ethnicity` have been changed from `"No"` and `"African American"` to respectively `"Yes"` and `"Caucasian"`, which now come first in the frequency tables for `Married` and `Ethnicity`.

What happens to the model output after doing the releveling? Run CHUNK 10 to refit the full model.

```
# CHUNK 10
# To make sure factors in the training set are releveled
train <- Credit[partition, ]
model.full <- lm(Balance ~ . + Income:Student + Rating:Student, data = train)
summary(model.full)
```

```
##
## Call:
## lm(formula = Balance ~ . + Income:Student + Rating:Student, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -204.83  -67.69  -10.17   60.36  300.13
##
```

```
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   -553.80111   36.77148 -15.061  < 2e-16 ***
## Income                          -7.41624    0.27847 -26.633  < 2e-16 ***
## Rating                           3.93051    0.06245  62.936  < 2e-16 ***
## Cards                            5.52797    4.12314   1.341  0.18107
## Age                             -0.73763    0.33635  -2.193  0.02910 *
## Education                       -0.30429    1.81060  -0.168  0.86665
## GenderMale                      10.56953   11.31508   0.934  0.35103
## StudentYes                     235.25194   53.25361   4.418 1.41e-05 ***
## MarriedNo                        4.01562   11.86499   0.338  0.73528
## EthnicityAfrican American      -19.08945   14.06755  -1.357  0.17585
## EthnicityAsian                  -4.47632   13.93650  -0.321  0.74830
## Income:StudentYes               -2.47014    0.80176  -3.081  0.00226 **
## Rating:StudentYes                0.85412    0.20479   4.171 4.02e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 97.26 on 288 degrees of freedom
## Multiple R-squared:  0.9601,Adjusted R-squared:  0.9584
## F-statistic: 577.2 on 12 and 288 DF,  p-value: < 2.2e-16
```

Comparing the output in CHUNK 10 with that in CHUNK 8, we can see that the coefficient estimates of the dummy variables for `Married` and `Ethnicity` have changed, so have the p-values for `EthnicityAsian`. If you try to drop the most statistically insignificant feature, that is, the one with the highest p-value, then the choice of the baseline level of the categorical predictors may affect which feature to drop. This explains the importance of releveling.

Next step: Removing non-predictive features.

TASK 3: Select features

Some of the features may lack predictive power and result in overfitting. Determine which features should be retained. Use the stepAIC function (from the MASS package) to make this determination. When using this function, there are two decisions to make.

- Use direction = "backward" or direction = "forward"
- Use AIC (k = 2) or BIC (k = log(nrow(train)))

Make each decision based on the business problem.

At this stage, the full model has a total of $p = 11$ eleven features. Naturally, not all of these 11 features are predictive of `Balance`, our target variable. While including all of these features will improve its fit to the training data, this may make the model too complex in size and result in overfitting (i.e., poor fit to unseen test data). A sensible way to improve its predictive power (and interpretability) is to drop statistically insignificant features and retain genuinely predictive ones. This is the main task of the rest of this subsection.

Notice that it is not a good idea to drop multiple features at a time as two features may appear insignificant together, but the removal of one feature can lead to the other feature becoming important. Bear in mind that the p-value column in the model summary answers the question of whether the feature in question provides additional predictive value *in the presence of other features.* It is therefore advisable to drop *one and only one* feature in every round of the removal process and stop until all that remain are significant enough.

One more preparatory step: Binarization in R. Before implementing feature selection in R, it is desirable to explicitly binarize categorical predictors with three or more levels. Although the `lm()` function automatically converts categorical predictors into dummy variables behind the scenes, as we saw in CHUNKS 8 and 10, explicitly doing binarization can help make the feature selection process more meaningful. This is because feature selection functions in R like `drop1()` and `stepAIC()` (to be introduced below) treat factor variables as a single feature and either retain the variable with all of its existing levels or remove the variable completely. This "all or nothing" feature selection approach does not allow for the possibility that individual levels of a factor with three or more levels may be insignificant with regard to the base level (and hence could be combined with it) or insignificantly different from other levels (in which case they could be combined into a new, bigger level).

Compared to generating new features such as `x^2`, `x^3`, `log(x)`, `exp(x)`, `sin(x)` for a numeric variable `x`, binarization of categorical variables is conceptually more subtle and practically more difficult to implement in R. The PA modules suggest the use of the `dummyVars()` function in the `caret` package and we will follow suit. Among the four categorical variables `Gender`, `Student`, `Married`, and `Ethnicity`, only `Ethnicity` has more than two levels and has to be binarized by two dummy variables. The binarization of `Ethnicity` can be done by running CHUNK 11.

```
# CHUNK 11 (Binarization)
library(caret)
binarizer <- dummyVars(paste("~ ", paste(vars.categorical, collapse = "+")),
                       data = Credit, fullRank = TRUE)
# OR type out the categorical predictors one by one
#binarizer <- dummyVars(~ Gender + Student + Married + Ethnicity,
#                       data = Credit, fullRank = TRUE)
binarized_vars <- data.frame(predict(binarizer, newdata = Credit))
head(binarized_vars)

##   Gender.Male Student.Yes Married.No Ethnicity.African.American
## 1           1           0          0                          0
## 2           0           1          0                          0
## 3           1           0          1                          0
## 4           0           0          1                          0
## 5           1           0          0                          0
## 6           1           0          1                          0
##   Ethnicity.Asian
## 1               0
## 2               1
## 3               1
## 4               1
```

```
## 5                0
## 6                0
```

The `dummyVars()` function takes a formula (there is no target variable) as the first argument and a data frame carrying the categorical variable(s) you want to binarize as the second argument. In the command `paste(vars.categorical, collapse = "+")`, we use the `paste()` function introduced in Section 1.3 to "paste" the names of the four categorical predictors `Gender`, `Student`, `Married`, and `Ethnicity` (recall that `vars.categorical` is defined in CHUNK 5) and separate them by the plus (`"+"`) sign. Then another application of the `paste()` function inserts the tilde sign before the "sum" of the names of the categorical predictors and forms the overall formula. (The use of the `paste()` function is suggested in the PA e-learning modules and projects. If you find the `paste()` function too difficult to work with, you may simply type out the formula `"~ Gender + Student + Married + Ethnicity"`.) The `fullRank` option allows you to choose whether a full rank or less than full rank parameterization of the model matrix should be used, if the option is set to `TRUE` (appropriate for regression) or `FALSE` (the default value and more appropriate for principal components and cluster analyses; see Chapter 6), respectively. By setting `fullRank = TRUE`, the baseline levels of the categorical predictors are left out; otherwise, the baseline levels would be retained and a rank-deficient model would result.

```
# A repeat of CHUNK 11 with fullRank = FALSE
binarizer <- dummyVars(paste("~ ", paste(vars.categorical, collapse = "+")),
                       data = Credit, fullRank = FALSE)
binarized_vars <- data.frame(predict(binarizer, newdata = Credit))
head(binarized_vars)

##   Gender.Female Gender.Male Student.No Student.Yes Married.Yes Married.No
## 1             0           1          1           0           1          0
## 2             1           0          0           1           1          0
## 3             0           1          1           0           0          1
## 4             1           0          1           0           0          1
## 5             0           1          1           0           1          0
## 6             0           1          1           0           0          1
##   Ethnicity.Caucasian Ethnicity.African.American Ethnicity.Asian
## 1                   1                          0               0
## 2                   0                          0               1
## 3                   0                          0               1
## 4                   0                          0               1
## 5                   1                          0               0
## 6                   1                          0               0
```

As we said above, it is enough to binarize only `Ethnicity` as it is the only categorical predictor with three or more levels. For illustration purposes, we have binarized all of the four categorical predictors to show you how *multiple* multi-level factors can be dealt with; this is likely the case you will encounter in the exam. As we expect, the binarization procedure creates two dummy variables for `Ethnicity`, called `Ethnicity.African American` and `Ethnicity.Asian` (recall that the baseline level after doing the releveling is `Caucasian`), indicating whether `Ethnicity` equals `African American` or `Asian`, respectively.

The `binarized_vars` data frame contains all of the dummy variables. In CHUNK 12, let's combine it with the original `Credit` dataset via the `cbind()` function (introduced on page 49) and rename it with a `.bin` suffix to distinguish it from the original non-binarized dataset.

```
# CHUNK 12
Credit.bin <- cbind(Credit, binarized_vars)
head(Credit.bin)

##    Income Rating Cards Age Education Gender Student Married Ethnicity
## 1  14.891    283     2  34        11   Male      No     Yes Caucasian
## 2 106.025    483     3  82        15 Female     Yes     Yes     Asian
## 3 104.593    514     4  71        11   Male      No      No     Asian
## 4 148.924    681     3  36        11 Female      No      No     Asian
## 5  55.882    357     2  68        16   Male      No     Yes Caucasian
## 6  80.180    569     4  77        10   Male      No      No Caucasian
##   Balance Gender.Male Student.Yes Married.No Ethnicity.African.American
## 1     333           1           0          0                          0
## 2     903           0           1          0                          0
## 3     580           1           0          1                          0
## 4     964           0           0          1                          0
## 5     331           1           0          0                          0
## 6    1151           1           0          1                          0
##   Ethnicity.Asian
## 1               0
## 2               1
## 3               1
## 4               1
## 5               0
## 6               0
```

Given the binarized variables, the original categorical predictors are no longer needed as they duplicate the information contained in the dummy variables. They can be dropped by assigning them to the `NULL` symbol.

```
# CHUNK 12 (Cont.)
Credit.bin$Gender <- NULL
Credit.bin$Student <- NULL
Credit.bin$Married <- NULL
Credit.bin$Ethnicity <- NULL
head(Credit.bin)

##    Income Rating Cards Age Education Balance Gender.Male Student.Yes
## 1  14.891    283     2  34        11     333           1           0
## 2 106.025    483     3  82        15     903           0           1
## 3 104.593    514     4  71        11     580           1           0
## 4 148.924    681     3  36        11     964           0           0
## 5  55.882    357     2  68        16     331           1           0
## 6  80.180    569     4  77        10    1151           1           0
```

```
##   Married.No Ethnicity.African.American Ethnicity.Asian
## 1          0                          0               0
## 2          0                          0               1
## 3          1                          0               1
## 4          1                          0               1
## 5          0                          0               0
## 6          1                          0               0
```

Now that the expanded dataset with binarized variables is available, we again split it into the training and test sets, using the same `partition` vector obtained earlier. There is no need to call the `createDataPartition()` function again as `partition` already contains the set of observations (rows) that should be sent to the training and test sets, and the same training/test split should be used to evaluate all of our candidate models.

```
# CHUNK 12 (Cont.)
train.bin <- Credit.bin[partition, ]
test.bin <- Credit.bin[-partition, ]
```

You may ask:

Is it necessary to remember the R code above for doing binarization?

In both the June 2019 exam and the Hospital Readmissions sample project that the SOA released in May 2019, a code chunk for performing binarization of factor variables is provided to help you. You merely have to change the placeholders to names of the variables in the dataset you are working on. Here is how it looks.

```
library(caret)

# insert the names of the variables to be binarized
factor_names <- c("<var1>", "<var2>", ...)
factor_vars <- <data>[, factor_names]

# dummyVars is not compatible with factors
for (var in factor_names) {
  factor_vars[, var] <- as.character(factor_vars[, var])
}

binarizer <- caret::dummyVars(paste("~ ", paste(factor_names, collapse = "+")),
                              data = factor_vars, fullRank = TRUE)
binarized_vars <- data.frame(predict(binarizer, newdata = factor_vars))
head(binarized_vars)
```

Notice two features of the SOA's code not present in our illustrative code given in CHUNK 11.

1. The first part of the SOA's code converts the factor variables to character variables. The PA modules and exams say that this conversion is done because "`dummyVars` is not compatible with factors," which does not seem true. Without this conversion, the binarization continues to work well, as we saw in CHUNK 11 above, and the linear models can still be fit properly, as we will see soon.

2. The command `caret:dummyVars` explicitly specifies that the `dummyVars()` function from the `caret` package is used.

It is very likely that future exams that require the use of binarization will also provide a code chunk for you. (If not, type `?dummyVars` to seek help. Remember to load the `caret` package first!)

We are now ready to do feature selection in earnest. In R, dropping insignificant features sequentially can be achieved in several ways, two of which are:

Feature selection by the `drop1()` function. The `drop1()` function allows you to conduct backward selection and exhibits the AIC of the current model together with the AIC that would result if one and only one of the features were to be dropped, hence the name "drop1."

In CHUNK 13, we use the `drop1()` function to see which feature in the *non-binarized* full model (constructed in CHUNK 8) should be removed first.

```
# CHUNK 13
drop1(model.full)

## Single term deletions
##
## Model:
## Balance ~ Income + Rating + Cards + Age + Education + Gender +
##     Student + Married + Ethnicity + Income:Student + Rating:Student
##                 Df Sum of Sq     RSS    AIC
## <none>                       2724393 2768.3
## Cards            1     17004 2741397 2768.2
## Age              1     45496 2769889 2771.3
## Education        1       267 2724660 2766.3
## Gender           1      8254 2732647 2767.2
## Married          1      1084 2725476 2766.4
## Ethnicity        2     17551 2741944 2766.2
## Income:Student   1     89791 2814184 2776.1
## Rating:Student   1    164545 2888938 2784.0
```

The model corresponding to `<none>` is the current model, where "none" of the variables are dropped. We can see that among all of the predictors, dropping `Ethnicity` decreases the AIC the most, from 2768.3 to 2766.2, and so `Ethnicity` should be dropped first (remember, the smaller the AIC, the better). We can then rerun the model without `Ethnicity` and repeat the process until none of the remaining features are worth removing. Do note that `Ethnicity` is associated with two degrees of freedom, one for each of the two non-baseline levels, `African American` and `Asian`. Without performing binarization in advance, the `drop1()` function will treat `Ethnicity` as a single predictor and drop the two dummy variables altogether, which ignores the possibility that only one of the two non-baseline levels is insignificant while the other one is significant (with respect to the baseline).

Now run CHUNK 14 to construct the binarized full model (on the `train.bin` data frame), print a summary of the model, and apply the `drop1()` function.

```
# CHUNK 14
model.full.bin <- lm(Balance ~ . + Income:Student.Yes + Rating:Student.Yes,
```

```
                    data = train.bin)
summary(model.full.bin)

##
## Call:
## lm(formula = Balance ~ . + Income:Student.Yes + Rating:Student.Yes,
##     data = train.bin)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -204.83  -67.69  -10.17   60.36  300.13
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  -553.80111   36.77148 -15.061  < 2e-16 ***
## Income                         -7.41624    0.27847 -26.633  < 2e-16 ***
## Rating                          3.93051    0.06245  62.936  < 2e-16 ***
## Cards                           5.52797    4.12314   1.341  0.18107
## Age                            -0.73763    0.33635  -2.193  0.02910 *
## Education                      -0.30429    1.81060  -0.168  0.86665
## Gender.Male                    10.56953   11.31508   0.934  0.35103
## Student.Yes                   235.25194   53.25361   4.418 1.41e-05 ***
## Married.No                      4.01562   11.86499   0.338  0.73528
## Ethnicity.African.American    -19.08945   14.06755  -1.357  0.17585
## Ethnicity.Asian                -4.47632   13.93650  -0.321  0.74830
## Income:Student.Yes             -2.47014    0.80176  -3.081  0.00226 **
## Rating:Student.Yes              0.85412    0.20479   4.171 4.02e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 97.26 on 288 degrees of freedom
## Multiple R-squared:  0.9601,Adjusted R-squared:  0.9584
## F-statistic: 577.2 on 12 and 288 DF,  p-value: < 2.2e-16

drop1(model.full.bin)

## Single term deletions
##
## Model:
## Balance ~ Income + Rating + Cards + Age + Education + Gender.Male +
##     Student.Yes + Married.No + Ethnicity.African.American + Ethnicity.Asian +
##     Income:Student.Yes + Rating:Student.Yes
##                          Df Sum of Sq     RSS    AIC
## <none>                                 2724393 2768.3
## Cards                     1     17004 2741397 2768.2
## Age                       1     45496 2769889 2771.3
## Education                 1       267 2724660 2766.3
```

```
## Gender.Male                  1     8254 2732647 2767.2
## Married.No                   1     1084 2725476 2766.4
## Ethnicity.African.American   1    17419 2741812 2768.2
## Ethnicity.Asian              1      976 2725369 2766.4
## Income:Student.Yes           1    89791 2814184 2776.1
## Rating:Student.Yes           1   164545 2888938 2784.0
```

The summary output is identical to that in CHUNK 10, showing that binarization has no effect on the quality of the fitted model. However, when the `drop1()` function is applied, the two dummy variables for `Ethnicity` are now considered separately. If only one of the numeric predictors or dummy variables is omitted, then the one resulting in the greatest drop in the AIC is `Education`, but not any of the two dummy variables for `Ethnicity`. We can then drop `Education` and look for the next most insignificant feature.

As a side note, observe that the `drop1()` function (as well as the `stepAIC()` function we will learn next) respects the hierarchical principle. You can see that only the interaction terms `Income:Student.Yes` and `Rating:Student.Yes` are considered for dropping, but not the individual variables `Income`, `Rating`, and `Student.Yes`.

(Preferred!) Feature selection by the `stepAIC()` function. Available from the `MASS` package, the `stepAIC()` function automates stepwise model selection and implements the `drop1()` function sequentially, in each step adding or dropping the feature that causes the greatest improvement in the model according to a certain criterion (the AIC by default). The process is terminated until no more variables can be added or dropped to improve the model.

Not much about the `stepAIC()` function is presented in the PA e-learning modules, but it is used in all of the PA sample and exam projects (every PA project involves feature selection in some way!). The June 2019 PA exam even tests some of the options of the function that can be changed to increase the flexibility of the feature selection process, so it pays to look at this feature selection function closely.

When the `stepAIC()` function is used, we can set two important options to control the feature selection process and selection criterion:

- *Selection process*—`direction = "backward"` (default) or `"forward"`: With the `stepAIC()` function, we can either implement:
  - ▷ Backward selection, where we start with the full model (with interaction and polynomial terms inserted if desired) and in each step drop the feature that leads to the greatest decrease in the model AIC (or BIC).
  - ▷ Forward selection, where we start with a primitive model (usually the model with only the intercept) and add the feature, one at a time, that leads to the greatest decrease in the model AIC (or BIC).
- *Selection criterion*—`k = 2` or `k = log(nrow(train))`: Despite the nomenclature of the `stepAIC()` function, it can perform stepwise selection using not only the AIC (default), but also the BIC as the selection criterion. To change the selection criterion from the AIC to the BIC, adjust the `k` argument, which is the multiple of the number of degrees of freedom that serves as the penalty term. Since AIC $= -2l + \boxed{2}p$ and BIC $= -2l + \boxed{\ln(n)}p$, the AIC and BIC are captured by setting `k = 2` (default) and `k = log(nrow(train))`, respectively.

Run CHUNK 15 to perform stepwise backward selection using the AIC on the binarized full model, save the final model as an `lm` object named `model.backward` for future manipulation, and print a summary of the final model.

```
# CHUNK 15
#install.packages("MASS")  # uncomment this line the first time you use MASS
library(MASS)
model.backward <- stepAIC(model.full.bin)

## Start:  AIC=2768.3
## Balance ~ Income + Rating + Cards + Age + Education + Gender.Male +
##     Student.Yes + Married.No + Ethnicity.African.American + Ethnicity.Asian +
##     Income:Student.Yes + Rating:Student.Yes
##
##                               Df Sum of Sq     RSS    AIC
## - Education                    1       267 2724660 2766.3
## - Ethnicity.Asian              1       976 2725369 2766.4
## - Married.No                   1      1084 2725476 2766.4
## - Gender.Male                  1      8254 2732647 2767.2
## - Cards                        1     17004 2741397 2768.2
## - Ethnicity.African.American   1     17419 2741812 2768.2
## <none>                                     2724393 2768.3
## - Age                          1     45496 2769889 2771.3
## - Income:Student.Yes           1     89791 2814184 2776.1
## - Rating:Student.Yes           1    164545 2888938 2784.0
##
## Step:  AIC=2766.33
## Balance ~ Income + Rating + Cards + Age + Gender.Male + Student.Yes +
##     Married.No + Ethnicity.African.American + Ethnicity.Asian +
##     Income:Student.Yes + Rating:Student.Yes
##
##                               Df Sum of Sq     RSS    AIC
## - Ethnicity.Asian              1      1045 2725705 2764.4
## - Married.No                   1      1103 2725763 2764.5
## - Gender.Male                  1      8155 2732815 2765.2
## - Cards                        1     17094 2741754 2766.2
## - Ethnicity.African.American   1     17600 2742260 2766.3
## <none>                                     2724660 2766.3
## - Age                          1     45305 2769965 2769.3
## - Income:Student.Yes           1     89550 2814210 2774.1
## - Rating:Student.Yes           1    164305 2888965 2782.0
##
## Step:  AIC=2764.45
## Balance ~ Income + Rating + Cards + Age + Gender.Male + Student.Yes +
##     Married.No + Ethnicity.African.American + Income:Student.Yes +
##     Rating:Student.Yes
##
```

```
##                                Df Sum of Sq     RSS    AIC
## - Married.No                    1      1259 2726964 2762.6
## - Gender.Male                   1      8349 2734054 2763.4
## - Ethnicity.African.American    1     16662 2742367 2764.3
## - Cards                         1     16801 2742506 2764.3
## <none>                                      2725705 2764.4
## - Age                           1     45195 2770900 2767.4
## - Income:Student.Yes            1     88669 2814374 2772.1
## - Rating:Student.Yes            1    163417 2889122 2780.0
##
## Step:  AIC=2762.59
## Balance ~ Income + Rating + Cards + Age + Gender.Male + Student.Yes +
##     Ethnicity.African.American + Income:Student.Yes + Rating:Student.Yes
##
##                                Df Sum of Sq     RSS    AIC
## - Gender.Male                   1      8654 2735618 2761.5
## - Ethnicity.African.American    1     16083 2743048 2762.4
## - Cards                         1     17506 2744470 2762.5
## <none>                                      2726964 2762.6
## - Age                           1     44184 2771148 2765.4
## - Income:Student.Yes            1     90501 2817466 2770.4
## - Rating:Student.Yes            1    167204 2894168 2778.5
##
## Step:  AIC=2761.54
## Balance ~ Income + Rating + Cards + Age + Student.Yes + Ethnicity.African.American +
##     Income:Student.Yes + Rating:Student.Yes
##
##                                Df Sum of Sq     RSS    AIC
## - Ethnicity.African.American    1     16455 2752073 2761.3
## - Cards                         1     17641 2753259 2761.5
## <none>                                      2735618 2761.5
## - Age                           1     45032 2780650 2764.5
## - Income:Student.Yes            1     93630 2829248 2769.7
## - Rating:Student.Yes            1    170632 2906251 2777.8
##
## Step:  AIC=2761.35
## Balance ~ Income + Rating + Cards + Age + Student.Yes + Income:Student.Yes +
##     Rating:Student.Yes
##
##                          Df Sum of Sq     RSS    AIC
## - Cards                   1     17287 2769361 2761.2
## <none>                                2752073 2761.3
## - Age                     1     47526 2799599 2764.5
## - Income:Student.Yes      1     98062 2850136 2769.9
## - Rating:Student.Yes      1    173001 2925074 2777.7
##
## Step:  AIC=2761.23
```

```
## Balance ~ Income + Rating + Age + Student.Yes + Income:Student.Yes +
##     Rating:Student.Yes
##
##                        Df Sum of Sq     RSS    AIC
## <none>                               2769361 2761.2
## - Age                   1     44411 2813771 2764.0
## - Income:Student.Yes    1     98347 2867708 2769.7
## - Rating:Student.Yes    1    175231 2944592 2777.7

summary(model.backward)

##
## Call:
## lm(formula = Balance ~ Income + Rating + Age + Student.Yes +
##     Income:Student.Yes + Rating:Student.Yes, data = train.bin)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -198.713  -68.905   -3.523   59.063  289.741
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -542.28051   23.17499 -23.399  < 2e-16 ***
## Income                -7.45135    0.27482 -27.114  < 2e-16 ***
## Rating                 3.93723    0.06158  63.940  < 2e-16 ***
## Age                   -0.72304    0.33299  -2.171  0.03071 *
## Student.Yes          232.28427   52.84056   4.396 1.54e-05 ***
## Income:Student.Yes    -2.56507    0.79384  -3.231  0.00137 **
## Rating:Student.Yes     0.87034    0.20179   4.313 2.20e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 97.05 on 294 degrees of freedom
## Multiple R-squared:  0.9594,Adjusted R-squared:  0.9586
## F-statistic:  1159 on 6 and 294 DF,  p-value: < 2.2e-16
```

When executed, `stepAIC()` prints out the results of all iterations, showing how the model AIC (shown in the `AIC` column in ascending order) changes when the most insignificant feature is dropped in each step. For instance, in the first iteration dropping `Education` leads to the lowest model AIC, consistent with what we got in CHUNK 14 using the `drop1()` function. Features are dropped in the following order: `Education`, `Ethnicity.Asian`, `Married.No`, `Gender.Male`, `Ethnicity.African.American`, and `Cards`. Only the following six features remain in the final model:

- `Income`
- `Rating`
- `Age`

- `Student.Yes`
- `Income:Student.Yes`
- `Rating:Student.Yes`

All of the coefficient estimates in the final model are statistically significant and the retention of these six features is in agreement with what we got in Task 1.

EXAM NOTE

The June 2019 PA exam model solution says that "[c]andidates should run the final model" after performing stepwise model selection and "check that everything looks right." You can make simple comments like how many or which features are in the final model, whether their retention matches what you observed in the data exploration part, and whether the coefficient estimates are statistically significant (not always the case).

To perform forward selection requires more coding effort. Besides specifying `direction = "forward"`, in the `scope` argument of the `stepAIC()` function we use a two-component list to define the range of models to be examined in the stepwise selection process. The most sophisticated model, often the full model with all of the features, is given in the `upper` component while the most primitive model, usually the null model, is given in the `lower` component. Now run CHUNK 16 to select the final model based on forward selection (and the AIC), save it as an `lm` object named `model.forward`, and print a summary.

```
# CHUNK 16
# first fit the null model (i.e., model with no predictors)
model.null <- lm(Balance ~ 1, data = train.bin)

model.forward <- stepAIC(model.null, direction = "forward",
                         scope = list(upper = model.full.bin,
                                      lower = model.null))

## Start:  AIC=3713.8
## Balance ~ 1
##
##                                Df Sum of Sq      RSS    AIC
## + Rating                        1  52118308 16130877 3281.6
## + Income                        1  16318500 51930685 3633.6
## + Student.Yes                   1   4815192 63433993 3693.8
## + Cards                         1    630541 67618644 3713.0
## <none>                                      68249185 3713.8
## + Gender.Male                   1     95205 68153981 3715.4
## + Married.No                    1     83052 68166134 3715.4
## + Age                           1     80005 68169181 3715.4
## + Ethnicity.African.American    1       422 68248763 3715.8
## + Ethnicity.Asian               1       387 68248799 3715.8
```

```
## + Education                          1       235 68248951 3715.8
##
## Step:  AIC=3281.63
## Balance ~ Rating
##
##                                   Df Sum of Sq      RSS    AIC
## + Income                           1   8477509  7653369 3059.2
## + Student.Yes                      1   4454246 11676631 3186.4
## + Age                              1    408692 15722186 3275.9
## + Married.No                       1    109061 16021816 3281.6
## <none>                                         16130877 3281.6
## + Ethnicity.African.American       1    105708 16025169 3281.7
## + Cards                            1     90760 16040117 3281.9
## + Education                        1     17131 16113746 3283.3
## + Gender.Male                      1     15381 16115496 3283.3
## + Ethnicity.Asian                  1     10848 16120029 3283.4
##
## Step:  AIC=3059.21
## Balance ~ Rating + Income
##
##                                   Df Sum of Sq     RSS    AIC
## + Student.Yes                      1   4650209 3003160 2779.6
## + Age                              1     95929 7557440 3057.4
## + Married.No                       1     82760 7570608 3057.9
## <none>                                         7653369 3059.2
## + Ethnicity.Asian                  1     33142 7620226 3059.9
## + Ethnicity.African.American       1     15649 7637719 3060.6
## + Education                        1     11275 7642093 3060.8
## + Gender.Male                      1      4288 7649080 3061.0
## + Cards                            1       806 7652562 3061.2
##
## Step:  AIC=2779.63
## Balance ~ Rating + Income + Student.Yes
##
##                                   Df Sum of Sq     RSS    AIC
## + Rating:Student.Yes               1     83132 2920027 2773.2
## + Age                              1     57608 2945552 2775.8
## + Ethnicity.African.American       1     20915 2982245 2779.5
## <none>                                         3003160 2779.6
## + Cards                            1     15989 2987171 2780.0
## + Gender.Male                      1     13881 2989279 2780.2
## + Ethnicity.Asian                  1      3777 2999383 2781.2
## + Married.No                       1      2863 3000297 2781.3
## + Income:Student.Yes               1       921 3002239 2781.5
## + Education                        1       237 3002923 2781.6
##
## Step:  AIC=2773.18
```

```
## Balance ~ Rating + Income + Student.Yes + Rating:Student.Yes
##
##                                Df Sum of Sq     RSS    AIC
## + Income:Student.Yes            1    106256 2813771 2764.0
## + Age                           1     52320 2867708 2769.7
## + Ethnicity.African.American    1     23651 2896376 2772.7
## <none>                                      2920027 2773.2
## + Cards                         1     14186 2905842 2773.7
## + Gender.Male                   1     13792 2906236 2773.8
## + Ethnicity.Asian               1      1494 2918534 2775.0
## + Married.No                    1      1401 2918626 2775.0
## + Education                     1         5 2920022 2775.2
##
## Step:  AIC=2764.02
## Balance ~ Rating + Income + Student.Yes + Rating:Student.Yes +
##     Income:Student.Yes
##
##                                Df Sum of Sq     RSS    AIC
## + Age                           1     44411 2769361 2761.2
## <none>                                      2813771 2764.0
## + Ethnicity.African.American    1     18521 2795250 2764.0
## + Cards                         1     14172 2799599 2764.5
## + Gender.Male                   1     10042 2803729 2764.9
## + Married.No                    1       322 2813449 2766.0
## + Ethnicity.Asian               1       208 2813563 2766.0
## + Education                     1       141 2813631 2766.0
##
## Step:  AIC=2761.23
## Balance ~ Rating + Income + Student.Yes + Age + Rating:Student.Yes +
##     Income:Student.Yes
##
##                                Df Sum of Sq     RSS    AIC
## <none>                                      2769361 2761.2
## + Cards                         1   17287.2 2752073 2761.3
## + Ethnicity.African.American    1   16101.2 2753259 2761.5
## + Gender.Male                   1    9158.6 2760202 2762.2
## + Married.No                    1    1522.7 2767838 2763.1
## + Education                     1     410.7 2768950 2763.2
## + Ethnicity.Asian               1      82.0 2769279 2763.2

summary(model.forward)

##
## Call:
## lm(formula = Balance ~ Rating + Income + Student.Yes + Age +
##     Rating:Student.Yes + Income:Student.Yes, data = train.bin)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -198.713  -68.905   -3.523   59.063  289.741
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -542.28051   23.17499 -23.399  < 2e-16 ***
## Rating               3.93723    0.06158  63.940  < 2e-16 ***
## Income              -7.45135    0.27482 -27.114  < 2e-16 ***
## Student.Yes        232.28427   52.84056   4.396 1.54e-05 ***
## Age                 -0.72304    0.33299  -2.171  0.03071 *
## Rating:Student.Yes   0.87034    0.20179   4.313 2.20e-05 ***
## Income:Student.Yes  -2.56507    0.79384  -3.231  0.00137 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 97.05 on 294 degrees of freedom
## Multiple R-squared:  0.9594,Adjusted R-squared:  0.9586
## F-statistic:  1159 on 6 and 294 DF,  p-value: < 2.2e-16
```

Features are added in the following order: `Rating`, `Income`, `Student.Yes`, `Rating:Student.Yes`, `Income:Student.Yes`, `Age`. In this case, the final model based on forward selection turns out to be identical to that based on backward selection.

Example 3.4.1. (Feature selection using the BIC) In the `Credit` data, determine the final models based on backward and forward selections, using the BIC as the selection criterion. How do these models compare to those based on the AIC?

Solution. To determine the final models with BIC being the selection criterion, we can modify the code in CHUNKS 15 and 16 and add the option `k = log(nrow(train.bin))` in the `stepAIC()` function. Run CHUNK 17 to see the output. Notice that the AIC column carries, in fact, the values of the BIC for different linear models. To check this, recall from CHUNK 15 that the AIC of the full model is 2,768.3. As AIC $= -2l + 2p$ and the full model has $p = 13$ regression coefficients, the loglikelihood of the full model is $l = -1,371.15$. It follows that its BIC is $-2l + (\ln n)p = -2(-1,371.15) + (\ln 301)(13) = 2,816.5$, as we see in the first line of the output in CHUNK 17.

Like the AIC, the BIC-based models resulting from backward selection and forward selection are the same, both containing `Income`, `Rating`, `Student.Yes`, `Income:Student.Yes`, and `Rating:Student.Yes`. Compared to the final model based on the AIC, the final model based on the BIC is smaller in size, with only five features; `Age` is not included. □

```
# CHUNK 17 (Example 3.4.1)
model.backward.BIC <- stepAIC(model.full.bin, k = log(nrow(train.bin)))

## Start:  AIC=2816.5
```

```
## Balance ~ Income + Rating + Cards + Age + Education + Gender.Male +
##     Student.Yes + Married.No + Ethnicity.African.American + Ethnicity.Asian +
##     Income:Student.Yes + Rating:Student.Yes
##
##                              Df Sum of Sq     RSS    AIC
## - Education                   1       267 2724660 2810.8
## - Ethnicity.Asian             1       976 2725369 2810.9
## - Married.No                  1      1084 2725476 2810.9
## - Gender.Male                 1      8254 2732647 2811.7
## - Cards                       1     17004 2741397 2812.7
## - Ethnicity.African.American  1     17419 2741812 2812.7
## - Age                         1     45496 2769889 2815.8
## <none>                                    2724393 2816.5
## - Income:Student.Yes          1     89791 2814184 2820.6
## - Rating:Student.Yes          1    164545 2888938 2828.4
##
## Step:  AIC=2810.82
## Balance ~ Income + Rating + Cards + Age + Gender.Male + Student.Yes +
##     Married.No + Ethnicity.African.American + Ethnicity.Asian +
##     Income:Student.Yes + Rating:Student.Yes
##
##                              Df Sum of Sq     RSS    AIC
## - Ethnicity.Asian             1      1045 2725705 2805.2
## - Married.No                  1      1103 2725763 2805.2
## - Gender.Male                 1      8155 2732815 2806.0
## - Cards                       1     17094 2741754 2807.0
## - Ethnicity.African.American  1     17600 2742260 2807.1
## - Age                         1     45305 2769965 2810.1
## <none>                                    2724660 2810.8
## - Income:Student.Yes          1     89550 2814210 2814.8
## - Rating:Student.Yes          1    164305 2888965 2822.7
##
## Step:  AIC=2805.23
## Balance ~ Income + Rating + Cards + Age + Gender.Male + Student.Yes +
##     Married.No + Ethnicity.African.American + Income:Student.Yes +
##     Rating:Student.Yes
##
##                              Df Sum of Sq     RSS    AIC
## - Married.No                  1      1259 2726964 2799.7
## - Gender.Male                 1      8349 2734054 2800.4
## - Ethnicity.African.American  1     16662 2742367 2801.3
## - Cards                       1     16801 2742506 2801.4
## - Age                         1     45195 2770900 2804.5
## <none>                                    2725705 2805.2
## - Income:Student.Yes          1     88669 2814374 2809.2
## - Rating:Student.Yes          1    163417 2889122 2817.1
##
```

```
## Step:  AIC=2799.66
## Balance ~ Income + Rating + Cards + Age + Gender.Male + Student.Yes +
##     Ethnicity.African.American + Income:Student.Yes + Rating:Student.Yes
##
##                                Df Sum of Sq     RSS    AIC
## - Gender.Male                   1      8654 2735618 2794.9
## - Ethnicity.African.American    1     16083 2743048 2795.7
## - Cards                         1     17506 2744470 2795.9
## - Age                           1     44184 2771148 2798.8
## <none>                                      2726964 2799.7
## - Income:Student.Yes            1     90501 2817466 2803.8
## - Rating:Student.Yes            1    167204 2894168 2811.9
##
## Step:  AIC=2794.91
## Balance ~ Income + Rating + Cards + Age + Student.Yes + Ethnicity.African.American +
##     Income:Student.Yes + Rating:Student.Yes
##
##                                Df Sum of Sq     RSS    AIC
## - Ethnicity.African.American    1     16455 2752073 2791.0
## - Cards                         1     17641 2753259 2791.1
## - Age                           1     45032 2780650 2794.1
## <none>                                      2735618 2794.9
## - Income:Student.Yes            1     93630 2829248 2799.3
## - Rating:Student.Yes            1    170632 2906251 2807.4
##
## Step:  AIC=2791
## Balance ~ Income + Rating + Cards + Age + Student.Yes + Income:Student.Yes +
##     Rating:Student.Yes
##
##                          Df Sum of Sq     RSS    AIC
## - Cards                   1     17287 2769361 2787.2
## - Age                     1     47526 2799599 2790.4
## <none>                                2752073 2791.0
## - Income:Student.Yes      1     98062 2850136 2795.8
## - Rating:Student.Yes      1    173001 2925074 2803.7
##
## Step:  AIC=2787.18
## Balance ~ Income + Rating + Age + Student.Yes + Income:Student.Yes +
##     Rating:Student.Yes
##
##                          Df Sum of Sq     RSS    AIC
## - Age                     1     44411 2813771 2786.3
## <none>                                2769361 2787.2
## - Income:Student.Yes      1     98347 2867708 2792.0
## - Rating:Student.Yes      1    175231 2944592 2799.9
##
## Step:  AIC=2786.26
```

```
## Balance ~ Income + Rating + Student.Yes + Income:Student.Yes +
##     Rating:Student.Yes
##
##                       Df Sum of Sq     RSS    AIC
## <none>                             2813771 2786.3
## - Income:Student.Yes  1    106256 2920027 2791.7
## - Rating:Student.Yes  1    188468 3002239 2800.1

model.forward.BIC <- stepAIC(model.null, direction = "forward",
                        scope = list(upper = model.full.bin, lower = model.null),
                        k = log(nrow(train.bin)))

## Start:  AIC=3717.51
## Balance ~ 1
##
##                                Df Sum of Sq      RSS    AIC
## + Rating                        1  52118308 16130877 3289.0
## + Income                        1  16318500 51930685 3641.0
## + Student.Yes                   1   4815192 63433993 3701.2
## <none>                                      68249185 3717.5
## + Cards                         1    630541 67618644 3720.4
## + Gender.Male                   1     95205 68153981 3722.8
## + Married.No                    1     83052 68166134 3722.8
## + Age                           1     80005 68169181 3722.9
## + Ethnicity.African.American    1       422 68248763 3723.2
## + Ethnicity.Asian               1       387 68248799 3723.2
## + Education                     1       235 68248951 3723.2
##
## Step:  AIC=3289.04
## Balance ~ Rating
##
##                                Df Sum of Sq      RSS    AIC
## + Income                        1   8477509  7653369 3070.3
## + Student.Yes                   1   4454246 11676631 3197.5
## + Age                           1    408692 15722186 3287.0
## <none>                                      16130877 3289.0
## + Married.No                    1    109061 16021816 3292.7
## + Ethnicity.African.American    1    105708 16025169 3292.8
## + Cards                         1     90760 16040117 3293.1
## + Education                     1     17131 16113746 3294.4
## + Gender.Male                   1     15381 16115496 3294.5
## + Ethnicity.Asian               1     10848 16120029 3294.5
##
## Step:  AIC=3070.33
## Balance ~ Rating + Income
##
##                                Df Sum of Sq     RSS    AIC
```

```
## + Student.Yes                    1   4650209 3003160 2794.5
## <none>                                       7653369 3070.3
## + Age                            1     95929 7557440 3072.2
## + Married.No                     1     82760 7570608 3072.8
## + Ethnicity.Asian                1     33142 7620226 3074.7
## + Ethnicity.African.American     1     15649 7637719 3075.4
## + Education                      1     11275 7642093 3075.6
## + Gender.Male                    1      4288 7649080 3075.9
## + Cards                          1       806 7652562 3076.0
##
## Step:  AIC=2794.46
## Balance ~ Rating + Income + Student.Yes
##
##                                Df Sum of Sq     RSS    AIC
## + Rating:Student.Yes            1     83132 2920027 2791.7
## + Age                           1     57608 2945552 2794.3
## <none>                                      3003160 2794.5
## + Ethnicity.African.American    1     20915 2982245 2798.1
## + Cards                         1     15989 2987171 2798.6
## + Gender.Male                   1     13881 2989279 2798.8
## + Ethnicity.Asian               1      3777 2999383 2799.8
## + Married.No                    1      2863 3000297 2799.9
## + Income:Student.Yes            1       921 3002239 2800.1
## + Education                     1       237 3002923 2800.1
##
## Step:  AIC=2791.71
## Balance ~ Rating + Income + Student.Yes + Rating:Student.Yes
##
##                                Df Sum of Sq     RSS    AIC
## + Income:Student.Yes            1    106256 2813771 2786.3
## <none>                                      2920027 2791.7
## + Age                           1     52320 2867708 2792.0
## + Ethnicity.African.American    1     23651 2896376 2795.0
## + Cards                         1     14186 2905842 2795.9
## + Gender.Male                   1     13792 2906236 2796.0
## + Ethnicity.Asian               1      1494 2918534 2797.3
## + Married.No                    1      1401 2918626 2797.3
## + Education                     1         5 2920022 2797.4
##
## Step:  AIC=2786.26
## Balance ~ Rating + Income + Student.Yes + Rating:Student.Yes +
##     Income:Student.Yes
##
##                                Df Sum of Sq     RSS    AIC
## <none>                                      2813771 2786.3
## + Age                           1     44411 2769361 2787.2
## + Ethnicity.African.American    1     18521 2795250 2790.0
```

```
## + Cards                1    14172 2799599 2790.4
## + Gender.Male          1    10042 2803729 2790.9
## + Married.No           1      322 2813449 2791.9
## + Ethnicity.Asian      1      208 2813563 2791.9
## + Education            1      141 2813631 2792.0
```

AIC vs. BIC, backward vs. forward: Which ones to use? When performing stepwise feature selection, which selection criterion and selection process to use is an important modeling decision. Naturally, none of the selection criteria and selection processes are universally superior and an argument in favor of a particular criterion or process can be put forward based on the context of the given business problem.

- *AIC vs. BIC:* Recall that the AIC and BIC differ only in terms of the size of the penalty that applies to the number of parameters. For the AIC, the penalty is 2; for the BIC, the penalty is the natural logarithm of the size of the training set, or $\ln n_{\text{train}} = \ln 301 = 5.7071$, which is greater. For a more complex model to beat a less complex model in terms of the BIC, the goodness of fit of the former has to far surpass that of the latter to make up for the increase in the penalty term. Thus the BIC will tend to favor a model with fewer features and represents a more conservative approach to feature selection. Since our goal in this case study is to identify *key* factors that relate to the target variable `Balance`, it makes sense to take a conservative approach and work with as few features as necessary in the final model using the BIC as the selection criterion.

  You can also make an argument (though not as convincing) in support of the AIC, saying that due to its propensity for retaining more features than other selection criteria, the client you work with will be given more factors to consider and the risk of missing important features will be smaller.

- *Backward vs. forward selections:* The decision of using forward or backward selections can be justified with similar reasons. With the selection criterion held fixed, forward selection is more likely to produce a final model with fewer features and better aligns with our goal of identifying *key* factors affecting `Balance`.

EXAM NOTE

As the June 2019 exam model solution says, "[a]ny combination of criterion and selection process is valid *if justified appropriately*." However, "[b]eing more comfortable with one method, a method being the default, or a method being commonly used are not satisfactory justifications for selecting a method."

Based on the above considerations, `model.forward.BIC` will be our recommended model. In the next subsection, we will see that it also fares the best with respect to prediction accuracy.

3.4.3 Model Validation

TASK 4: Validate the model

Run the models resulting from stepwise selection in the preceding subsection, evaluate the RMSE against the test set, and compare it to the full model. Make a recommendation as to which model should be used. In addition, generate and interpret diagnostic plots for this model to check the model assumptions.

Which model is most predictive? Now that we have different linear models of varying degrees of complexity fitted to the training set, it is time to compare their predictive performance on the common *test* set and see which one is the most predictive. We are particularly interested in whether reducing the complexity of a linear model by stepwise selection algorithms, as we did in Subsection 3.4.2, can enhance prediction accuracy. As discussed in Subsection 3.2.2, a natural measure of the performance of linear models is the test RMSE. The smaller the test RMSE, the better. CHUNK 18 computes and prints the test RMSE of the four models we have constructed.

```
# CHUNK 18
# define the rmse() function again
rmse <- function(observed, predicted) {
  sqrt(mean((observed - predicted)^2))
}

rmse(test$Balance, predict(model.null, newdata = test.bin))

## [1] 403.7726

rmse(test$Balance, predict(model.full.bin, newdata = test.bin))

## [1] 108.8133

rmse(test$Balance, predict(model.backward, newdata = test.bin))

## [1] 108.3417

rmse(test$Balance, predict(model.backward.BIC, newdata = test.bin))

## [1] 107.8142
```

For your convenience, the following table reports the test RMSE extracted from the R output for each of the four models.

Model	Test RMSE
Null model (benchmark)	403.7726
Full model	108.8133
Final model based on AIC	108.3417
Final model based on BIC	107.8142 (smallest)

EXAM NOTE

The December 2018 PA exam solution says that "[c]andidates seldom presented an easy-to-see comparison of model results to assess validation methodology." Using a table like the one shown above can help.

Observe that the two reduced models resulting from stepwise selection have a (slightly) lower[xxv] test RMSE than the full model despite having far fewer features. Comparing the two reduced models, we can see that the model based on the BIC not only is simpler in the sense that it is nested within that based on the AIC, but also is more predictive in the sense that its test RMSE is lower. On the basis of both prediction accuracy and interpretability, the final model based on the BIC is justifiably the most promising model among all of the linear models we have considered.

Model diagnostics. Now that the model based on the BIC appears to be the best among the four linear models, let's diagnose it carefully for any anomalies with the use of appropriate graphical displays, as in CHUNK 19.

When the `plot()` function (this is one of the rare instances in Exam PA in which we make use of R's base graphics functions) is applied to an `lm` object (or `glm` object) such as `model.backward.BIC`, four diagnostic plots are generated, the first two of which are most easily interpreted.

- The "Residuals vs Fitted" plot graphs the residuals of the model against the fitted values on the training set. If the fitted model has sufficiently captured the true relationship between the target variable and the predictors, then there should be no discernible patterns in this plot, where the residuals are dispersed in an erratic manner. The plot is also an informal test of the homogeneity of error variance. If the amount of fluctuation of the residuals increases as the magnitude of the fitted values increases, exhibiting a *funnel shape*, then the homoscedasticity assumption is at stake.

  Unfortunately, while the homoscedastic assumption does not seem to be violated, there is a clear U-shape in the "Residuals vs Fitted" plot, suggesting that there are nonlinear patterns beyond interaction effects that fail to be captured by the fitted model. One may try to refine the model by incorporating polynomial terms such as $\texttt{Income}^2$ and $\texttt{Rating}^2$. (Exercise: Try this and see how much the prediction accuracy will improve!)

- The "Normal Q-Q" plot graphs the quantiles of the standardized residuals (which are the residuals divided by their standard errors) against the theoretical standard normal quantiles and can be used for checking the normality assumption. If the points lie closely on a 45° straight line passing through the origin, then the normality assumption is supported.

  In CHUNK 19, the q-q plot shows that the normal distribution assumption is maintained for most values, but not in the two extreme tails. This suggests using a response distribution with a fatter tail than the normal distribution, the topic of Chapter 4.

[xxv] If you change the random seed in CHUNK 7 (where we did the training/test split), you may run into a situation where the two reduced models have a slightly higher test RMSE than the full model. In this case, you can still argue in favor of the reduced models saying that the predictive performance of the reduced models is "comparable" to the full model, but they are much simpler and easier to interpret.

```
# CHUNK 19
plot(model.backward.BIC)
```

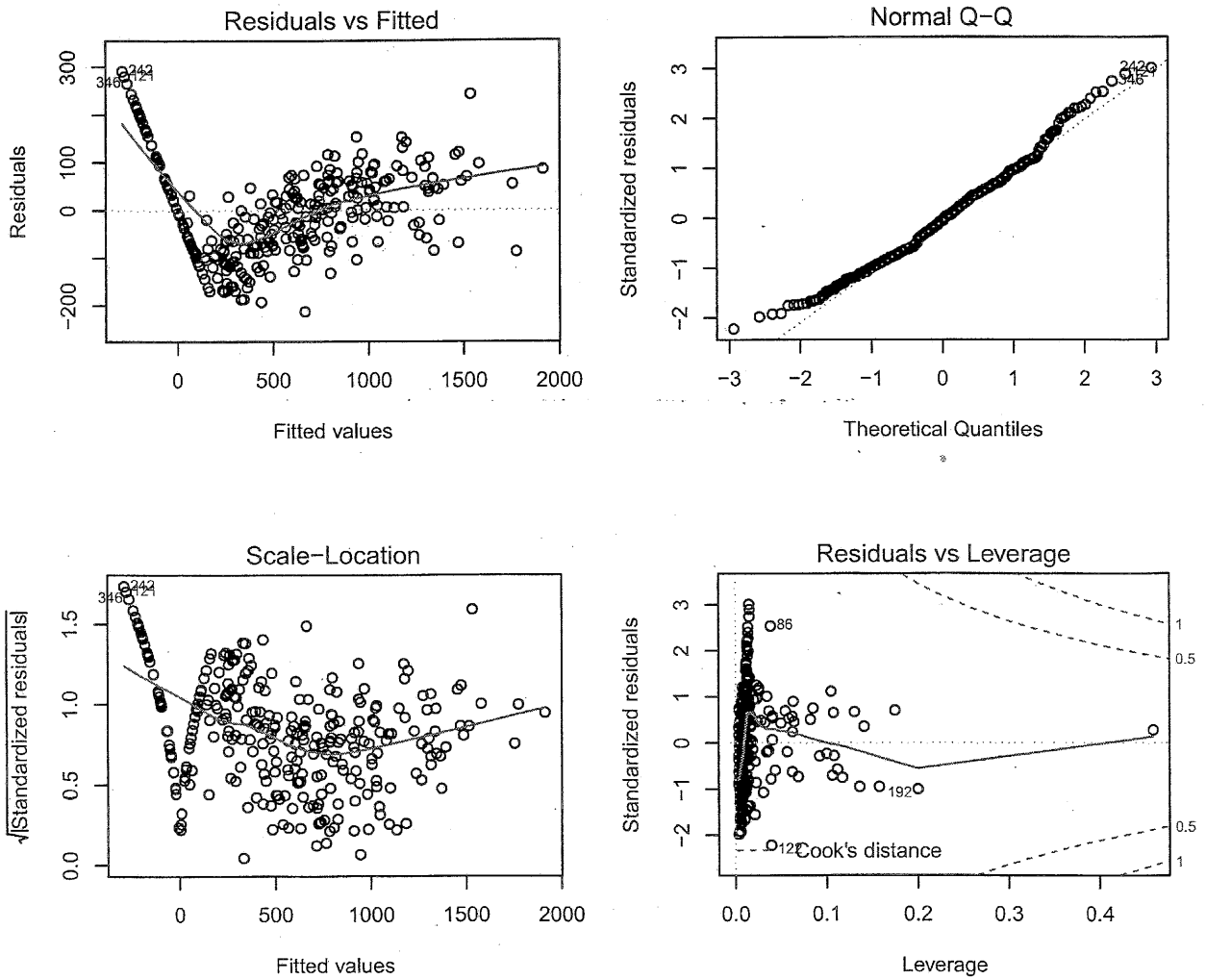


Figure 3.4.6: Diagnostic plots for the final model based on the BIC fitted to the training set of the `Credit` data.

EXAM NOTE

The December 2018 PA exam solution says that after fitting and selecting a linear model, "[f]ew candidates performed the valuable step of error analysis," although they were not explicitly told to do so in the project statement. "[T]hose who did so mostly used q-q plots or residual plots." Of course, there is nothing wrong with q-q plots or residual plots!

3.4.4 Regularization

TASK 5: Investigate ridge regression and lasso

Decide whether to run ridge regression or lasso and compare the test RMSE of the model of your choice with that of the recommended model in Subsection 3.4.3. Which model would you recommend for this analysis? Do not base your recommendation solely on the RMSE from each model.

An alternative to using forward or backward selection for identifying useful features is regularization, which reduces the magnitude of the coefficient estimates via the use of a penalty term and serves to prevent overfitting. Variables with limited predictive power will receive a small, if not exactly zero, coefficient estimate and are thus removed from the model. In this subsection, we will fit a lasso regression model (due to its variable selection property and our preference for key factors affecting `Balance`) to the `Credit` dataset and compare its performance to that of the models based on the stepwise model selection in Subsection 3.4.2.

Fitting penalized regression models: `glmnet()` function. To implement ridge regression and the lasso, we will use the `glmnet` package (available on Exam PA), where the main function is the `glmnet()` function. This function can be used to fit ridge regression models, lasso models, and general elastic nets, but it has different syntax from the `lm()` function we have been using. Instead of directly inputting a formula of the form `Y ~ X1 + ... + Xp`, we have to pass in a data matrix `x` (recall the design matrix defined in (3.2.4)) carrying the values of the features and a response vector `y`. The design matrix can be created using the `model.matrix()` function from a given formula and a data frame. Run CHUNK 20 to set up the design matrix for the training part of the `Credit` dataset and look at the first six rows. The formula should involve all of the potentially useful features, including the two interaction variables, not just the predictive features identified earlier.

```
# CHUNK 20
X.train <- model.matrix(Balance ~ . + Rating:Student + Income:Student,
                        data = train)
head(X.train)  # print out a few rows of the design matrix
```

```
##    (Intercept)  Income Rating Cards Age Education GenderMale StudentYes
## 1            1  14.891    283     2  34        11          1          0
## 3            1 104.593    514     4  71        11          1          0
## 4            1 148.924    681     3  36        11          0          0
## 5            1  55.882    357     2  68        16          1          0
```

```
## 10           1  71.061    491     3  41          19           0           1
## 13           1  80.616    394     1  57           7           0           0
##    MarriedNo EthnicityAfrican American EthnicityAsian Rating:StudentYes
## 1          0                         0              0                 0
## 3          1                         0              1                 0
## 4          1                         0              1                 0
## 5          0                         0              0                 0
## 10         0                         1              0               491
## 13         0                         0              1                 0
##    Income:StudentYes
## 1              0.000
## 3              0.000
## 4              0.000
## 5              0.000
## 10            71.061
## 13             0.000
```

A particular advantage of the `model.matrix()` function is that categorical predictors will be automatically transformed into dummy variables, just like the `dummyVars()` function,[xxvi] and the design matrix is of full rank. This advance conversion is important because unlike the `lm()` function, the `glmnet()` function can only work on quantitative inputs, but not character variables. In other words, explicit binarization of categorical variables is required for the `glmnet()` function to work. As an example, you can see that `Ethnicity` has been automatically converted into two dummy variables corresponding to "`African American`" or "`Asian`" (recall that "`Caucasian`" is the baseline level of `Ethnicity` after releveling).

As soon as a design matrix has been set up, it can be passed to the `glmnet()` function along with the vector of response values and the values of the hyperparameters α and λ. Run CHUNK 21[xxvii] to apply the lasso (corresponding to $\alpha = 1$, which is the default value of `alpha`) to the training set with $\lambda = 10^0 = 1, 10^1 = 10, 10^2 = 100, 10^3 = 1,000$ (passed as the vector `10^(0:3)`) and save the fitted model as an object named `lasso`. Note that the `family` argument is set to `"gaussian"` because we are dealing with linear models with normal errors.

```
# CHUNK 21
#install.packages("glmnet")  # uncomment this line the first time you use glmnet
library(glmnet)
lasso <- glmnet(x = X.train, y = train$Balance, family = "gaussian",
                lambda = 10^(0:3), alpha = 1)
```

To appreciate the effect of regularization, it is informative to extract the coefficient estimates and see how they vary with the value of λ. This can be done in two ways.

1. *(Suggested in the PA e-learning modules)* A `glmnet` object is a list, two components of which are:

[xxvi] Although the code for `dummyVars()` function is more complex than that of `model.matrix()`, the former by default produces a design matrix of less than full rank, which is more appropriate for doing principal components and cluster analyses.

[xxvii] You may get slightly different output depending on which version of the `glmnet` package and/or R you are using.

`a0` This is a vector of size `length(lambda)` hosting the value of the intercept estimate, one for each value of λ.

`beta` This is a matrix of coefficient estimates for the predictors, excluding the intercept, arranged by columns according to different values of λ.

Then we use two separate commands to access the estimated intercept and slope coefficients.

```
# CHUNK 21 (Cont.)
# first way to extract coefficient estimates
lasso$a0

##        s0        s1        s2        s3
##  531.1362 -183.3551 -516.8619 -555.1325

lasso$beta

## 13 x 4 sparse Matrix of class "dgCMatrix"
##                                s0        s1          s2          s3
## (Intercept)                     .         .           .           .
## Income                          .         .  -6.4047003  -7.3563171
## Rating                          . 1.9563912   3.6787565   3.9137820
## Cards                           .         .   0.1275813   4.9974186
## Age                             .         .  -0.3299576  -0.7005179
## Education                       .         .           .           .
## GenderMale                      .         .           .   8.6338520
## StudentYes                      .         . 265.3799583 250.1347442
## MarriedNo                       .         .           .   3.0059538
## EthnicityAfrican American       .         .  -0.1034776 -16.3731282
## EthnicityAsian                  .         .           .  -0.7407126
## Rating:StudentYes               . 0.2380456   0.3450852   0.7266730
## Income:StudentYes               .         .           .  -1.8914534
```

2. *(Preferred and suggested in ISLR)* Applying the `coef()` function to a `glmnet` object returns a matrix of coefficient estimates, including the intercept, arranged by columns according to different values of λ.

```
# CHUNK 21 (Cont.)
# second way
coef(lasso)

## 14 x 4 sparse Matrix of class "dgCMatrix"
##                                s0          s1          s2          s3
## (Intercept)           531.1362 -183.3550989 -516.8618738 -555.1324995
## (Intercept)                 .           .           .           .
## Income                      .           .  -6.4047003  -7.3563171
## Rating                      .   1.9563912   3.6787565   3.9137820
```

```
## Cards                          .          .          0.1275813    4.9974186
## Age                            .          .         -0.3299576   -0.7005179
## Education                      .          .          .            .
## GenderMale                     .          .          .            8.6338520
## StudentYes                     .          .        265.3799583  250.1347442
## MarriedNo                      .          .          .            3.0059538
## EthnicityAfrican American      .          .         -0.1034776  -16.3731282
## EthnicityAsian                 .          .          .           -0.7407126
## Rating:StudentYes              .          0.2380456  0.3450852    0.7266730
## Income:StudentYes              .          .          .           -1.8914534

mean(train$Balance)

## [1] 531.1362
```

Regardless of which way you use to access the coefficient estimates, the values of λ are arranged in descending order across the four columns, with `s0`, `s1`, `s2`, and `s3` representing $\lambda = 1{,}000, 100, 10$, and 1, respectively. Dots in the output represent features whose coefficient estimates are exactly zero and thus dropped from the model. We can see that as λ increases from 1, 10, 100, to 1,000, more and more coefficient estimates shrink to zero, an illustration of the feature selection property of the lasso.

- When $\lambda = 1$, the coefficient estimates are similar to the least squares, unpenalized estimates in CHUNK 8 (page 184). Only `Education` is dropped due to its statistical insignificance.
- When $\lambda = 10$, more features such as `Education`, `GenderMale`, `MarriedNo`, `EthnicityAsian`, and `Income:StudentYes` are eliminated.
- When $\lambda = 100$, only `Rating` and `Rating:StudentYes` are retained. Note that the `glmnet()` function, unlike `drop1()` or `stepAIC()`, does not follow the hierarchical principle since `Rating:StudentYes` is retained, but not `StudentYes`.)
- When $\lambda = 1{,}000$, the regularization penalty is so large that all of the coefficient estimates except the intercept become exactly zero, in which case the model is simply the null model with only the intercept. The estimated intercept is $\hat{\beta}_0^L = 531.1362$, which is also the mean value of `Balance` in the training set, as you can check using the command `mean(train$Balance)`.

Hyperparameter tuning: `cv.glmnet()` function. The `glmnet()` function requires that value(s) of λ be supplied *a priori* and does not directly tell us the optimal value of λ. Fortunately, the companion function `cv.glmnet()` (with "cv" standing for "cross-validation") fills this gap. This function performs k-fold cross-validation for regularized regression models, calculates the cross-validation error, and produces the optimal value of λ as a by-product. By default, the function performs 10-fold cross-validation, but the number of cross-validation folds can be changed by setting the `nfolds` argument to another positive integer.

The `cv.glmnet()` function is commonly accompanied with an application of the `plot()` function to a `cv.glmnet` object. This returns a cross-validation curve, that is, the cross-validation error plotted as a function of the values of λ used. Such a plot has several useful features:

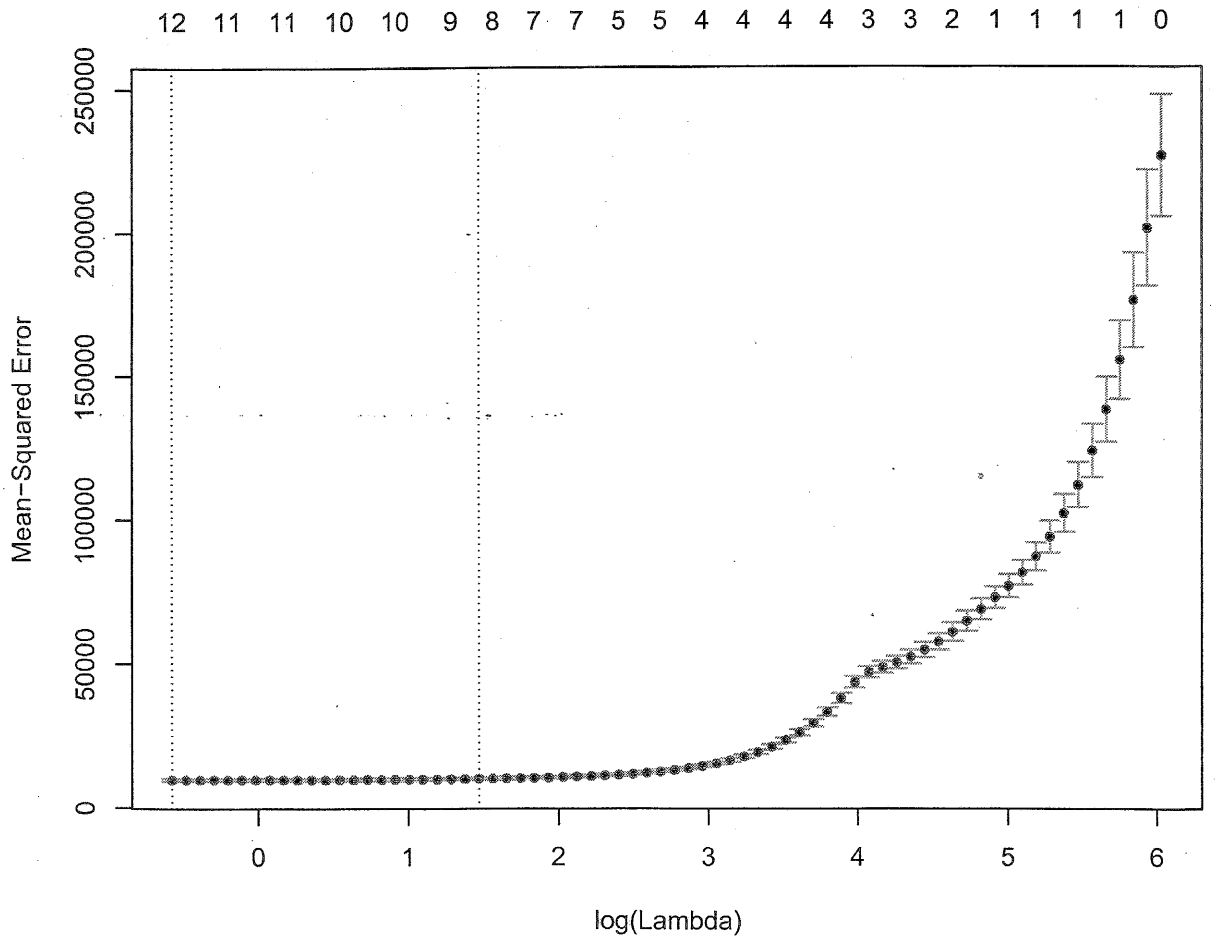
- The red dots represent the cross-validation error corresponding to different values of λ (or $\log \lambda$) and the vertical lines around the dots are the confidence interval bands.[xxviii]
- The numbers at the top of the graph indicate how many features have nonzero coefficient estimates at a given value of λ (or $\log \lambda$).
- The first vertical dotted line from the left corresponds to the value of λ (or $\log \lambda$) at which the cross-validation error is minimized. To directly extract such an optimal value of λ, we can look at the `lambda.min` component of a `cv.glmnet` object, which is a list.
- The second vertical dotted line is not discussed in the PA e-learning modules. It has to do with the *one-standard-error rule*, which you may have come across when studying for Exam SRM. In the current context, the rule corresponds to the greatest value of λ, or equivalently, the simplest regularized regression model such that the resulting cross-validation error is within one standard error of the minimum error (remember, the higher the value of λ, the more regularization there is and the simpler the model becomes). This value of λ is contained in the `lambda.1se` component of a `cv.glmnet` object.

  Although the one-standard-error rule is not presented in PA e-learning modules, it is covered in Exam SRM and is assumed knowledge for Exam PA. You can certainly make a comment on this rule in your write-up in Exam PA if you have time.

Now let's run CHUNK 22 to plot the cross-validation error curve for our `Credit` data, shown in Figure 3.4.7. Note that a random seed is set prior to the call of the `cv.glmnet()` function to make the choice of the cross-validation folds reproducible (this random seed need not be the same as that for the `createDataPartition()` function in CHUNK 7). The optimal value of λ is 0.5630216. Such a small value means that hardly any regularization is done to shrink the coefficient estimates. In fact, none of the coefficient estimates is set to zero.

[xxviii] Assuming normality, these confidence interval bands are given by cross-validation error $\pm$ 1.96 $\times$ standard error.

```
# CHUNK 22
set.seed(1111)
m <- cv.glmnet(x = X.train, y = train$Balance,
               family = "gaussian", alpha = 1)
plot(m)
```




```
m$lambda.min

## [1] 0.5630216
```

Figure 3.4.7: The cross-validation error curve for the lasso fitted to the training set of the `Credit` data.

We now refit the lasso model on the training set using the best value of λ found by cross-validation, save it as an object named `lasso.best`, make predictions on the test set, and finally evaluate the model's RMSE. These are all done in CHUNK 23. Notice that when the `predict()` function is applied to a `glmnet` object, the `newx` argument should be used instead of the `newdata` argument. To learn more, type `?predict.glmnet`.

```
# CHUNK 23
# Fit the lasso using the best value of lambda
lasso.best <- glmnet(x = X.train, y = train$Balance, family = "gaussian",
                     lambda = m$lambda.min, alpha = 1)

# Look at the coefficient estimates
lasso.best$beta

## 13 x 1 sparse Matrix of class "dgCMatrix"
##                                       s0
## (Intercept)                            .
## Income                        -7.3739775
## Rating                         3.9185005
## Cards                          5.2233269
## Age                           -0.7141180
## Education                     -0.1011369
## GenderMale                     9.4128722
## StudentYes                   237.8787238
## MarriedNo                      3.3217738
## EthnicityAfrican. American -17.6213968
## EthnicityAsian                -2.5916152
## Rating:StudentYes              0.8047331
## Income:StudentYes             -2.2025826

# Set up the design matrix for the test set
X.test <- model.matrix(Balance ~ . + Rating:Student + Income:Student,
                       data = test)

# Make predictions on the test set and calculate test RMSE
lasso.best.pred <- predict(lasso.best, newx = X.test)
rmse(test$Balance, lasso.best.pred)

## [1] 110.1028
```

The test RMSE of the lasso model is slightly higher than that of the final model found by the BIC (which is 107.8142) in Subsection 3.4.2. In addition to inferior predictive performance, the lasso model is also much harder to interpret due to its substantially larger size. As a result, the model based on the BIC remains the recommended model.

Let's end this subsection on regularization with two remarks:

1. *(Ridge regression)* Thus far we have set `alpha = 1` and performed the lasso. To implement ridge regression, we can set `alpha = 0` and repeat CHUNKS 21 to 23.

2. *(Optimal value of α)* Although the `cv.glmnet()` function allows us to pick the best value of λ, it has no built-in mechanism for choosing the best value of α. We can resort to trial and error, performing cross-validation over a grid of values of α and picking the value of α that produces the smallest cross-validation error. A `for` loop can be used for this purpose (look at CHUNK 7 of Section 6.7 of the PA modules if you are interested).