Question: Used Phones & Tablets Pricing Dataset

EAS 508 Assignment 3– Aditya Srivatsav Lolla and 50559685

```
Notes on R:
-   For the elastic net model, what we called λ in the videos, glmnet calls “alph
a”; you can get a range of results by varying alpha from 1 (lasso) to 0 (ridge regr
ession) [and, of course, other values of alpha in between].

-   In a function call like glmnet(x,y,family=”mgaussian”,alpha=1) the predictors x
need to be in R’s matrix format, rather than data frame format.  You can convert a
data frame to a matrix using as.matrix – for example, x <- as.matrix(data[,1:n-1])

- Rather than specifying a value of T, glmnet returns models for a variety of value
s of T.
```

```
library(ggplot2)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(MASS)
library(car)
```

```
## Loading required package: carData
```

```
library(class)
library(leaps)
library(glmnet)
```

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

```
## Loaded glmnet 4.1-8
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

Question: Used Phones & Tablets Pricing Dataset

The used and refurbished device market has grown considerably over the past decade as it provide cost-effective alternatives to both consumers and businesses that are looking to save money when purchasing one. Maximizing the longevity of devices through second-hand trade also reduces their environmental impact and helps in recycling and reducing waste. Here is a sample dataset of normalized used and new pricing data of refurbished / used devices.

- device_brand: Name of manufacturing brand
- os: OS on which the device runs
- screen_size: Size of the screen in cm
- 4g: Whether 4G is available or not
- 5g: Whether 5G is available or not
- front_camera_mp: Resolution of the rear camera in megapixels
- back_camera_mp: Resolution of the front camera in megapixels
- internal_memory: Amount of internal memory (ROM) in GB
- ram: Amount of RAM in GB
- battery: Energy capacity of the device battery in mAh
- weight: Weight of the device in grams
- release_year: Year when the device model was released
- days_used: Number of days the used/refurbished device has been used
- normalized_new_price: Normalized price of a new device of the same model
- normalized_used_price (response variable): Normalized price of the used/refurbished device

Read the data and answer the questions below:

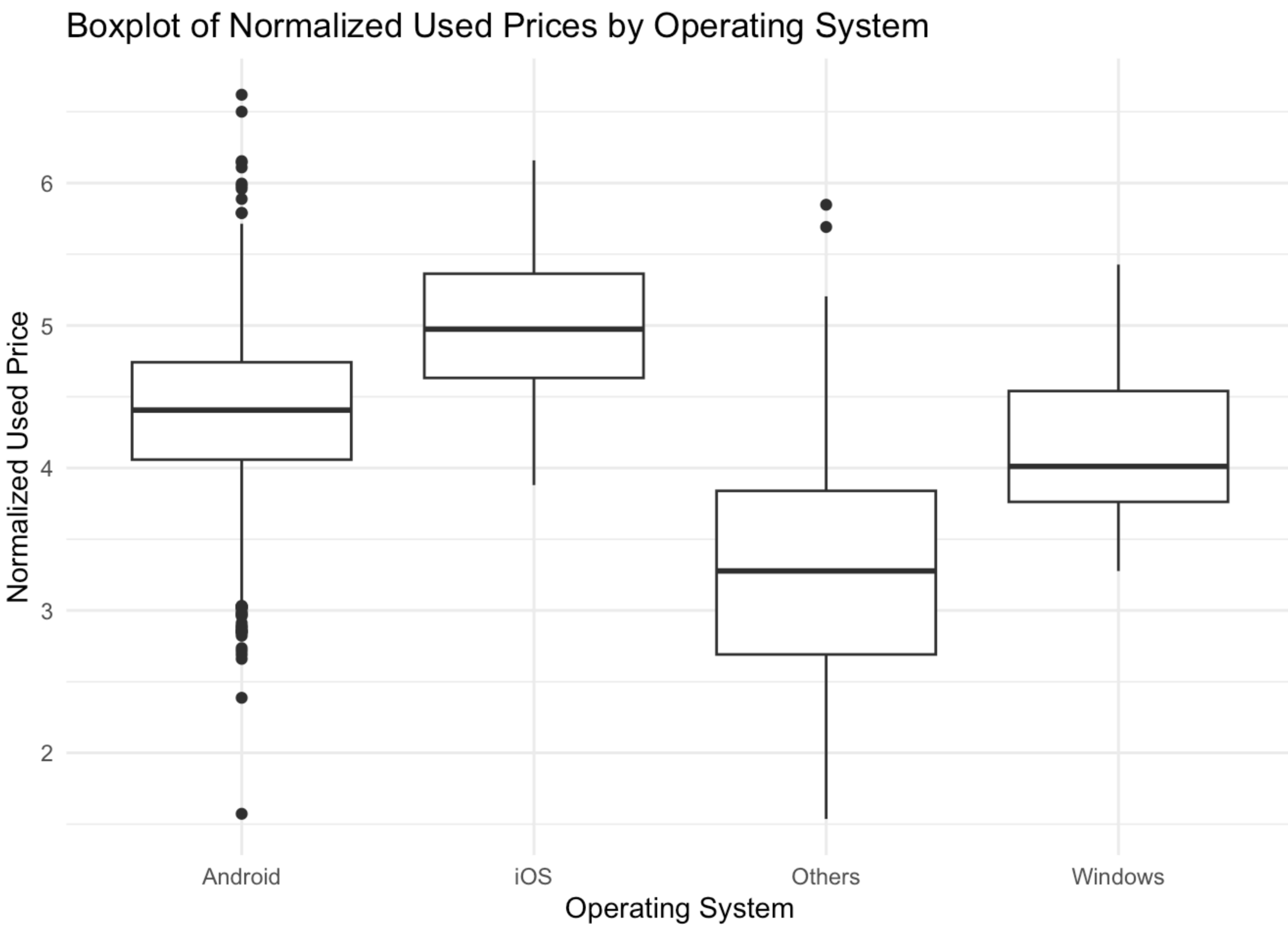
```
# Loading of the data
set.seed(100)
used_devices= read.csv("used_device_data.csv", header=TRUE, sep=",")

used_devices$device_brand=as.factor(used_devices$device_brand)
used_devices$os=as.factor(used_devices$os)
used_devices$X4g=as.factor(used_devices$X4g)
used_devices$X5g=as.factor(used_devices$X5g)

#Dividing the dataset into training and testing datasets
testRows = sample(nrow(used_devices),0.2*nrow(used_devices))
testData = used_devices[testRows, ]
trainData = used_devices[-testRows, ]
row.names(trainData) <- NULL
head(trainData)
```

```
##   device_brand      os screen_size X4g X5g rear_camera_mp front_camera_mp
## 1        Honor Android       14.50 yes  no             13               5
## 2        Honor Android       16.69 yes yes             13               8
## 3        Honor Android       15.32 yes  no             13               8
## 4        Honor Android       16.23 yes  no             13               8
## 5        Honor Android       13.84 yes  no              8               5
## 6        Honor Android       15.77 yes  no             13               8
##   internal_memory ram battery weight release_year days_used
## 1              64   3    3020    146         2020       127
## 2             128   8    4200    213         2020       162
## 3              64   3    5000    185         2020       293
## 4              64   4    4000    176         2020       223
## 5              32   2    3020    144         2020       234
## 6              64   4    3400    164         2020       219
##   normalized_new_price normalized_used_price
## 1             4.715100              4.307572
## 2             5.884631              5.111084
## 3             4.947837              4.389995
## 4             5.060694              4.413889
## 5             4.518958              3.878259
## 6             5.188726              4.729421
```

Part 1 [9 pts]: EXPLORATORY DATA ANALYSIS

a). (3 pts) Using trainData, create a boxplot of response variable “normalized_used_price” and “os”, with “normalized_used_price” on the vertical axis. Interpret the plot.Which os devices are the most expensive?

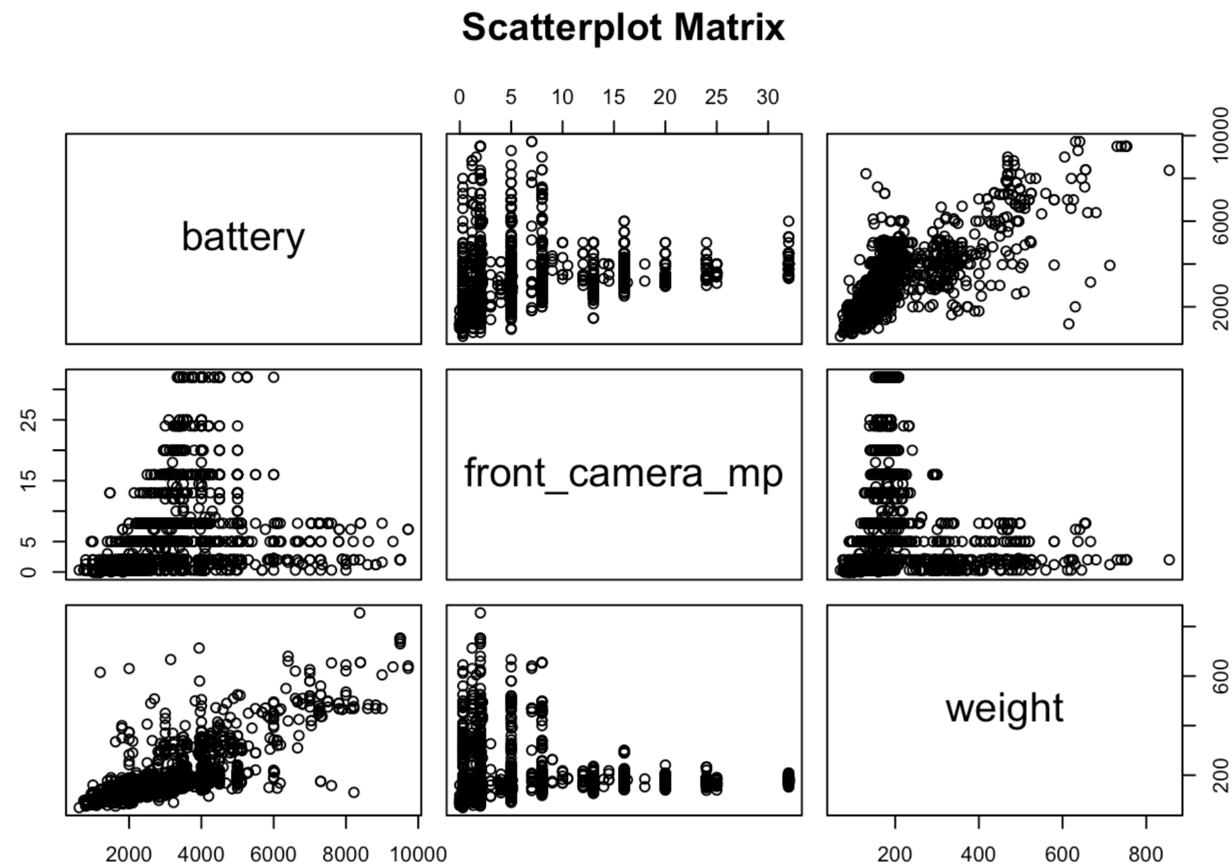
```
set.seed(100)
#BoxPlot Indicates the ggplot between Operating System and Normalized Used Prices
ggplot(trainData, aes(x = os, y = normalized_used_price)) +
  geom_boxplot() +
  labs(title = "Boxplot of Normalized Used Prices by Operating System",
       x = "Operating System",
       y = "Normalized Used Price") +
  theme_minimal()
```



Answer:

b). (6 pts) Using `trainData` , create a scatterplot matrix and a correlation table that includes the following continuous variables: `battery b) front_camera_mp c) weight` Does there appear to be multicollinearity among these three variables? Include your reasoning.

```
set.seed(100)
pairs(trainData[, c("battery", "front_camera_mp", "weight")],main = "Scatterplot Ma
trix")
```




```
correlation_matrix <- cor(trainData[, c("battery", "front_camera_mp", "weight")], u
se = "complete.obs")
correlation_matrix
```

```
##                   battery front_camera_mp       weight
## battery         1.0000000     0.345246272  0.734860874
## front_camera_mp 0.3452463     1.000000000 -0.007013192
## weight          0.7348609    -0.007013192  1.000000000
```

Answer:

Part 2 [8 pts]: MULTIPLE LINEAR REGRESSION

a). (6 pts) Create a multiple regression model using “normalized_used_price” as the response variable and all the predictors. Call it model1. Display the summary of the `model`.

- Which coefficients are statistically significant at the significance level of 0.05?
- Interpret the estimated coefficient of days_used and osWindows in the context of the problem. Mention any assumptions you make about other predictors clearly when stating the interpretation.

```
set.seed(100)
model1 <- lm(normalized_used_price ~ ., data = trainData)
summary(model1)
```

```
##
## Call:
## lm(formula = normalized_used_price ~ ., data = trainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.45262 -0.14092  0.02066  0.16708  1.11384
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -6.127e+01  8.913e+00  -6.874 7.83e-12 ***
## device_brandAlcatel    -3.031e-02  4.276e-02  -0.709 0.478532
## device_brandApple       4.284e-02  1.459e-01   0.294 0.769071
## device_brandAsus        4.908e-02  4.298e-02   1.142 0.253676
## device_brandBlackBerry  1.652e-01  7.443e-02   2.219 0.026554 *
## device_brandCelkon     -1.027e-01  5.748e-02  -1.786 0.074171 .
## device_brandCoolpad    -3.246e-02  7.419e-02  -0.437 0.661789
## device_brandGionee      1.489e-02  4.993e-02   0.298 0.765537
## device_brandGoogle     -4.739e-02  9.055e-02  -0.523 0.600755
## device_brandHonor       5.182e-03  4.405e-02   0.118 0.906349
## device_brandHTC        -1.830e-02  4.410e-02  -0.415 0.678175
## device_brandHuawei     -1.504e-02  3.994e-02  -0.377 0.706511
## device_brandKarbonn    -2.804e-02  6.569e-02  -0.427 0.669489
## device_brandLava        6.375e-03  5.579e-02   0.114 0.909048
## device_brandLenovo      2.218e-02  4.119e-02   0.539 0.590236
## device_brandLG         -4.473e-02  4.064e-02  -1.101 0.271101
## device_brandMeizu       1.965e-03  5.425e-02   0.036 0.971106
## device_brandMicromax   -3.233e-02  4.297e-02  -0.752 0.451908
## device_brandMicrosoft   5.044e-02  7.876e-02   0.640 0.521938
## device_brandMotorola   -3.513e-02  4.669e-02  -0.752 0.451911
## device_brandNokia       9.218e-02  4.678e-02   1.971 0.048875 *
## device_brandOnePlus     6.490e-03  1.104e-01   0.059 0.953140
## device_brandOppo        4.324e-03  4.360e-02   0.099 0.921019
## device_brandOthers     -8.846e-03  3.749e-02  -0.236 0.813496
## device_brandPanasonic  -1.004e-02  5.262e-02  -0.191 0.848710
## device_brandRealme      2.357e-01  1.106e-01   2.132 0.033124 *
## device_brandSamsung    -3.241e-02  3.880e-02  -0.835 0.403534
## device_brandSony       -5.723e-02  4.629e-02  -1.236 0.216415
## device_brandSpice      -1.989e-02  5.758e-02  -0.345 0.729766
## device_brandVivo       -3.697e-02  4.514e-02  -0.819 0.412866
## device_brandXiaomi      7.319e-02  4.434e-02   1.651 0.098892 .
## device_brandXOLO       -2.557e-02  5.238e-02  -0.488 0.625486
## device_brandZTE        -6.970e-03  4.295e-02  -0.162 0.871087
## osiOS                  -1.312e-01  1.465e-01  -0.896 0.370590
## osOthers               -1.044e-01  3.084e-02  -3.385 0.000724 ***
## osWindows              -1.720e-02  4.104e-02  -0.419 0.675165
## screen_size             2.152e-02  3.112e-03   6.913 5.97e-12 ***
## X4gyes                  3.558e-02  1.521e-02   2.340 0.019369 *
## X5gyes                 -4.694e-02  3.468e-02  -1.353 0.176082
## rear_camera_mp          2.101e-02  1.415e-03  14.845  < 2e-16 ***
## front_camera_mp         1.345e-02  1.101e-03  12.214  < 2e-16 ***
## internal_memory         5.927e-05  5.610e-05   1.057 0.290810
```

```
## ram                      2.391e-02  5.120e-03   4.670 3.17e-06 ***
## battery                 -2.171e-05  7.422e-06  -2.925 0.003472 **
## weight                   1.090e-03  1.223e-04   8.912  < 2e-16 ***
## release_year             3.100e-02  4.416e-03   7.020 2.84e-12 ***
## days_used                4.670e-05  2.927e-05   1.595 0.110771
## normalized_new_price     4.365e-01  1.187e-02  36.782  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2325 on 2555 degrees of freedom
## Multiple R-squared:  0.8471, Adjusted R-squared:  0.8442
## F-statistic: 301.1 on 47 and 2555 DF,  p-value: < 2.2e-16
```

Answer: the following coefficients are statistically significant at the 0.05 significance level (marked with *, **, or***):

Intercept device_brandBlackBerry device_brandNokia rear_camera_mp osOthers screen_size X4gyes front_camera_mp ram battery weight release_year rear_camera_mp normalized_new_price

b). (2 pts) Check model1 for multicollinearity using variance inflation factor (vif). Is multicollinearity a problem.Explain your conclusion.

```
set.seed(100)
vif_values <- vif(model1)
print(vif_values)
```

```
##                           GVIF Df GVIF^(1/(2*Df))
## device_brand         68.832293 32        1.068355
## os                   29.041436  3        1.753220
## screen_size           6.996924  1        2.645170
## X4g                   2.522135  1        1.588123
## X5g                   1.642238  1        1.281498
## rear_camera_mp        2.316113  1        1.521878
## front_camera_mp       2.684740  1        1.638518
## internal_memory       1.249868  1        1.117975
## ram                   2.063970  1        1.436652
## battery               4.580808  1        2.140282
## weight                5.775914  1        2.403313
## release_year          4.568995  1        2.137521
## days_used             2.328792  1        1.526038
## normalized_new_price  3.141529  1        1.772436
```

Answer:parts variables, such as screen_size, battery, weight, release_year, and os, exhibit multicollinearity, which can make parts of our model's predictions less dependable and more difficult to understand, according to the Variance Inflation Factor (VIF) values for our regression model. Particularly, the os variable exhibits the highest level of multicollinearity. We may need to eliminate some of these overlapping variables in order to improve our model. ____________

Part 3 [30 pts]: VARIABLE SELECTION

a. (12 pts) Conduct bestsubset selection, forward, backward step-wise regression on model1 using AIC (assume no controlling variables), and all the selected model `model2, model3, and model4` , respectively. Display the summary of the model. Note: Do not forget to put "trace=F" in order to prevent long printed outputs.

- What is the AIC and BIC of the selected model?
- Which of the original variables are selected?

```
You may use different methods other than the ones we used on the class. For exampl
e, using step(), see https://www.rdocumentation.org/packages/stats/versions/3.6.2/t
opics/step
```

```
#Best Model TepWise Regression
subset_best_mdel <- step(model1, direction = "both", trace = FALSE)
summary(subset_best_mdel)
```

```
##
## Call:
## lm(formula = normalized_used_price ~ device_brand + os + screen_size +
##     X4g + rear_camera_mp + front_camera_mp + ram + battery +
##     weight + release_year + days_used + normalized_new_price,
##     data = trainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.46685 -0.14223  0.02197  0.16668  1.13062
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -6.055e+01  8.801e+00  -6.880 7.50e-12 ***
## device_brandAlcatel    -3.033e-02  4.275e-02  -0.709 0.478119
## device_brandApple       5.730e-02  1.455e-01   0.394 0.693835
## device_brandAsus        5.110e-02  4.296e-02   1.189 0.234416
## device_brandBlackBerry  1.697e-01  7.438e-02   2.281 0.022623 *
## device_brandCelkon     -9.783e-02  5.652e-02  -1.731 0.083590 .
## device_brandCoolpad    -3.205e-02  7.420e-02  -0.432 0.665770
## device_brandGionee      1.580e-02  4.993e-02   0.317 0.751627
## device_brandGoogle     -4.317e-02  9.048e-02  -0.477 0.633289
## device_brandHonor       4.085e-03  4.400e-02   0.093 0.926044
## device_brandHTC        -1.786e-02  4.410e-02  -0.405 0.685459
## device_brandHuawei     -1.505e-02  3.990e-02  -0.377 0.705953
## device_brandKarbonn    -2.752e-02  6.563e-02  -0.419 0.675071
## device_brandLava        9.272e-03  5.559e-02   0.167 0.867556
## device_brandLenovo      2.282e-02  4.118e-02   0.554 0.579511
## device_brandLG         -4.533e-02  4.064e-02  -1.115 0.264749
## device_brandMeizu       2.996e-03  5.425e-02   0.055 0.955966
## device_brandMicromax   -3.103e-02  4.292e-02  -0.723 0.469842
## device_brandMicrosoft   4.978e-02  7.876e-02   0.632 0.527445
## device_brandMotorola   -3.428e-02  4.669e-02  -0.734 0.462891
## device_brandNokia       8.861e-02  4.672e-02   1.896 0.058009 .
## device_brandOnePlus     6.290e-03  1.104e-01   0.057 0.954589
## device_brandOppo        5.917e-03  4.360e-02   0.136 0.892062
## device_brandOthers     -8.222e-03  3.749e-02  -0.219 0.826447
## device_brandPanasonic  -9.958e-03  5.263e-02  -0.189 0.849931
## device_brandRealme      2.388e-01  1.106e-01   2.160 0.030851 *
## device_brandSamsung    -3.130e-02  3.879e-02  -0.807 0.419846
## device_brandSony       -5.757e-02  4.629e-02  -1.244 0.213716
## device_brandSpice      -1.897e-02  5.752e-02  -0.330 0.741565
## device_brandVivo       -3.377e-02  4.510e-02  -0.749 0.454084
## device_brandXiaomi      7.617e-02  4.429e-02   1.720 0.085565 .
## device_brandXOLO       -2.568e-02  5.238e-02  -0.490 0.623985
## device_brandZTE        -6.374e-03  4.295e-02  -0.148 0.882031
## osiOS                  -1.395e-01  1.464e-01  -0.953 0.340690
## osOthers               -1.089e-01  3.060e-02  -3.561 0.000377 ***
## osWindows              -1.602e-02  4.104e-02  -0.390 0.696237
## screen_size             2.154e-02  3.100e-03   6.949 4.65e-12 ***
## X4gyes                  3.546e-02  1.512e-02   2.346 0.019073 *
## rear_camera_mp          2.115e-02  1.390e-03  15.216  < 2e-16 ***
```

```
## front_camera_mp         1.356e-02  1.094e-03  12.386  < 2e-16 ***
## ram                     2.054e-02  4.614e-03   4.452 8.86e-06 ***
## battery                -2.151e-05  7.421e-06  -2.899 0.003780 **
## weight                  1.087e-03  1.222e-04   8.892  < 2e-16 ***
## release_year            3.065e-02  4.361e-03   7.028 2.68e-12 ***
## days_used               4.775e-05  2.919e-05   1.636 0.102016
## normalized_new_price    4.362e-01  1.142e-02  38.185  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2325 on 2557 degrees of freedom
## Multiple R-squared:  0.8469, Adjusted R-squared:  0.8442
## F-statistic: 314.3 on 45 and 2557 DF,  p-value: < 2.2e-16
```

```
AIC(subset_best_mdel)
```

```
## [1] -160.2183
```

```
BIC(subset_best_mdel)
```

```
## [1] 115.4094
```

```
coef(subset_best_mdel)
```

```
##            (Intercept)    device_brandAlcatel      device_brandApple
##          -6.055235e+01          -3.032950e-02           5.730179e-02
##       device_brandAsus device_brandBlackBerry     device_brandCelkon
##           5.109778e-02           1.696765e-01          -9.783200e-02
##    device_brandCoolpad     device_brandGionee     device_brandGoogle
##          -3.205464e-02           1.580418e-02          -4.317158e-02
##      device_brandHonor        device_brandHTC     device_brandHuawei
##           4.084860e-03          -1.786322e-02          -1.505475e-02
##    device_brandKarbonn       device_brandLava     device_brandLenovo
##          -2.751663e-02           9.271669e-03           2.282171e-02
##         device_brandLG      device_brandMeizu   device_brandMicromax
##          -4.533282e-02           2.996056e-03          -3.102629e-02
##  device_brandMicrosoft   device_brandMotorola      device_brandNokia
##           4.977796e-02          -3.427824e-02           8.860588e-02
##    device_brandOnePlus       device_brandOppo     device_brandOthers
##           6.289890e-03           5.916743e-03          -8.221584e-03
##  device_brandPanasonic     device_brandRealme    device_brandSamsung
##          -9.958189e-03           2.388479e-01          -3.130017e-02
##       device_brandSony      device_brandSpice       device_brandVivo
##          -5.757353e-02          -1.897179e-02          -3.377031e-02
##     device_brandXiaomi       device_brandXOLO        device_brandZTE
##           7.617136e-02          -2.568083e-02          -6.373548e-03
##                  osiOS               osOthers              osWindows
##          -1.394827e-01          -1.089367e-01          -1.602313e-02
##            screen_size                 X4gyes         rear_camera_mp
##           2.154112e-02           3.546300e-02           2.114567e-02
##        front_camera_mp                    ram                battery
##           1.355507e-02           2.054500e-02          -2.151235e-05
##                 weight           release_year              days_used
##           1.087018e-03           3.065151e-02           4.774726e-05
##   normalized_new_price
##           4.361640e-01
```

#Forward StepWise Regression

```
fcd_mdel <- step(model1, direction = "forward", trace = FALSE)
summary(fcd_mdel)
```

```
##
## Call:
## lm(formula = normalized_used_price ~ device_brand + os + screen_size +
##     X4g + X5g + rear_camera_mp + front_camera_mp + internal_memory +
##     ram + battery + weight + release_year + days_used + normalized_new_price,
##     data = trainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.45262 -0.14092  0.02066  0.16708  1.11384
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -6.127e+01  8.913e+00  -6.874 7.83e-12 ***
## device_brandAlcatel    -3.031e-02  4.276e-02  -0.709 0.478532
## device_brandApple       4.284e-02  1.459e-01   0.294 0.769071
## device_brandAsus        4.908e-02  4.298e-02   1.142 0.253676
## device_brandBlackBerry  1.652e-01  7.443e-02   2.219 0.026554 *
## device_brandCelkon     -1.027e-01  5.748e-02  -1.786 0.074171 .
## device_brandCoolpad    -3.246e-02  7.419e-02  -0.437 0.661789
## device_brandGionee      1.489e-02  4.993e-02   0.298 0.765537
## device_brandGoogle     -4.739e-02  9.055e-02  -0.523 0.600755
## device_brandHonor       5.182e-03  4.405e-02   0.118 0.906349
## device_brandHTC        -1.830e-02  4.410e-02  -0.415 0.678175
## device_brandHuawei     -1.504e-02  3.994e-02  -0.377 0.706511
## device_brandKarbonn    -2.804e-02  6.569e-02  -0.427 0.669489
## device_brandLava        6.375e-03  5.579e-02   0.114 0.909048
## device_brandLenovo      2.218e-02  4.119e-02   0.539 0.590236
## device_brandLG         -4.473e-02  4.064e-02  -1.101 0.271101
## device_brandMeizu       1.965e-03  5.425e-02   0.036 0.971106
## device_brandMicromax   -3.233e-02  4.297e-02  -0.752 0.451908
## device_brandMicrosoft   5.044e-02  7.876e-02   0.640 0.521938
## device_brandMotorola   -3.513e-02  4.669e-02  -0.752 0.451911
## device_brandNokia       9.218e-02  4.678e-02   1.971 0.048875 *
## device_brandOnePlus     6.490e-03  1.104e-01   0.059 0.953140
## device_brandOppo        4.324e-03  4.360e-02   0.099 0.921019
## device_brandOthers     -8.846e-03  3.749e-02  -0.236 0.813496
## device_brandPanasonic  -1.004e-02  5.262e-02  -0.191 0.848710
## device_brandRealme      2.357e-01  1.106e-01   2.132 0.033124 *
## device_brandSamsung    -3.241e-02  3.880e-02  -0.835 0.403534
## device_brandSony       -5.723e-02  4.629e-02  -1.236 0.216415
## device_brandSpice      -1.989e-02  5.758e-02  -0.345 0.729766
## device_brandVivo       -3.697e-02  4.514e-02  -0.819 0.412866
## device_brandXiaomi      7.319e-02  4.434e-02   1.651 0.098892 .
## device_brandXOLO       -2.557e-02  5.238e-02  -0.488 0.625486
## device_brandZTE        -6.970e-03  4.295e-02  -0.162 0.871087
## osiOS                  -1.312e-01  1.465e-01  -0.896 0.370590
## osOthers               -1.044e-01  3.084e-02  -3.385 0.000724 ***
## osWindows              -1.720e-02  4.104e-02  -0.419 0.675165
## screen_size             2.152e-02  3.112e-03   6.913 5.97e-12 ***
## X4gyes                  3.558e-02  1.521e-02   2.340 0.019369 *
## X5gyes                 -4.694e-02  3.468e-02  -1.353 0.176082
```

```
## rear_camera_mp          2.101e-02  1.415e-03  14.845  < 2e-16 ***
## front_camera_mp         1.345e-02  1.101e-03  12.214  < 2e-16 ***
## internal_memory         5.927e-05  5.610e-05   1.057 0.290810
## ram                     2.391e-02  5.120e-03   4.670 3.17e-06 ***
## battery                -2.171e-05  7.422e-06  -2.925 0.003472 **
## weight                  1.090e-03  1.223e-04   8.912  < 2e-16 ***
## release_year            3.100e-02  4.416e-03   7.020 2.84e-12 ***
## days_used               4.670e-05  2.927e-05   1.595 0.110771
## normalized_new_price    4.365e-01  1.187e-02  36.782  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2325 on 2555 degrees of freedom
## Multiple R-squared:  0.8471, Adjusted R-squared:  0.8442
## F-statistic: 301.1 on 47 and 2555 DF,  p-value: < 2.2e-16
```

```
AIC(fcd_mdel)
```

```
## [1] -158.9
```

```
BIC(fcd_mdel)
```

```
## [1] 128.4566
```

```
coef(fcd_mdel)
```

```
##            (Intercept)    device_brandAlcatel      device_brandApple
##          -6.126599e+01          -3.030502e-02           4.284209e-02
##       device_brandAsus device_brandBlackBerry     device_brandCelkon
##           4.907693e-02           1.651723e-01          -1.026801e-01
##    device_brandCoolpad     device_brandGionee     device_brandGoogle
##          -3.245876e-02           1.489093e-02          -4.739003e-02
##      device_brandHonor        device_brandHTC     device_brandHuawei
##           5.182494e-03          -1.830012e-02          -1.503909e-02
##    device_brandKarbonn       device_brandLava     device_brandLenovo
##          -2.804288e-02           6.374518e-03           2.218137e-02
##         device_brandLG      device_brandMeizu   device_brandMicromax
##          -4.473198e-02           1.965238e-03          -3.232779e-02
##  device_brandMicrosoft   device_brandMotorola      device_brandNokia
##           5.043973e-02          -3.512744e-02           9.217600e-02
##    device_brandOnePlus       device_brandOppo     device_brandOthers
##           6.489958e-03           4.323871e-03          -8.845737e-03
##  device_brandPanasonic     device_brandRealme    device_brandSamsung
##          -1.003953e-02           2.357450e-01          -3.241280e-02
##       device_brandSony      device_brandSpice       device_brandVivo
##          -5.723433e-02          -1.989134e-02          -3.697008e-02
##     device_brandXiaomi       device_brandXOLO        device_brandZTE
##           7.319137e-02          -2.556722e-02          -6.969922e-03
##                  osiOS               osOthers              osWindows
##          -1.312310e-01          -1.043887e-01          -1.720038e-02
##            screen_size                 X4gyes                 X5gyes
##           2.151577e-02           3.558147e-02          -4.693649e-02
##         rear_camera_mp        front_camera_mp        internal_memory
##           2.101284e-02           1.344500e-02           5.927443e-05
##                    ram                battery                 weight
##           2.390884e-02          -2.171048e-05           1.090351e-03
##           release_year              days_used   normalized_new_price
##           3.099896e-02           4.670001e-05           4.364681e-01
```

#BackWard Stepwise Regression

```
bcd_mdel <- step(model1, direction = "backward", trace = FALSE)
summary(bcd_mdel)
```

```
##
## Call:
## lm(formula = normalized_used_price ~ device_brand + os + screen_size +
##     X4g + rear_camera_mp + front_camera_mp + ram + battery +
##     weight + release_year + days_used + normalized_new_price,
##     data = trainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.46685 -0.14223  0.02197  0.16668  1.13062
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -6.055e+01  8.801e+00  -6.880 7.50e-12 ***
## device_brandAlcatel    -3.033e-02  4.275e-02  -0.709 0.478119
## device_brandApple       5.730e-02  1.455e-01   0.394 0.693835
## device_brandAsus        5.110e-02  4.296e-02   1.189 0.234416
## device_brandBlackBerry  1.697e-01  7.438e-02   2.281 0.022623 *
## device_brandCelkon     -9.783e-02  5.652e-02  -1.731 0.083590 .
## device_brandCoolpad    -3.205e-02  7.420e-02  -0.432 0.665770
## device_brandGionee      1.580e-02  4.993e-02   0.317 0.751627
## device_brandGoogle     -4.317e-02  9.048e-02  -0.477 0.633289
## device_brandHonor       4.085e-03  4.400e-02   0.093 0.926044
## device_brandHTC        -1.786e-02  4.410e-02  -0.405 0.685459
## device_brandHuawei     -1.505e-02  3.990e-02  -0.377 0.705953
## device_brandKarbonn    -2.752e-02  6.563e-02  -0.419 0.675071
## device_brandLava        9.272e-03  5.559e-02   0.167 0.867556
## device_brandLenovo      2.282e-02  4.118e-02   0.554 0.579511
## device_brandLG         -4.533e-02  4.064e-02  -1.115 0.264749
## device_brandMeizu       2.996e-03  5.425e-02   0.055 0.955966
## device_brandMicromax   -3.103e-02  4.292e-02  -0.723 0.469842
## device_brandMicrosoft   4.978e-02  7.876e-02   0.632 0.527445
## device_brandMotorola   -3.428e-02  4.669e-02  -0.734 0.462891
## device_brandNokia       8.861e-02  4.672e-02   1.896 0.058009 .
## device_brandOnePlus     6.290e-03  1.104e-01   0.057 0.954589
## device_brandOppo        5.917e-03  4.360e-02   0.136 0.892062
## device_brandOthers     -8.222e-03  3.749e-02  -0.219 0.826447
## device_brandPanasonic  -9.958e-03  5.263e-02  -0.189 0.849931
## device_brandRealme      2.388e-01  1.106e-01   2.160 0.030851 *
## device_brandSamsung    -3.130e-02  3.879e-02  -0.807 0.419846
## device_brandSony       -5.757e-02  4.629e-02  -1.244 0.213716
## device_brandSpice      -1.897e-02  5.752e-02  -0.330 0.741565
## device_brandVivo       -3.377e-02  4.510e-02  -0.749 0.454084
## device_brandXiaomi      7.617e-02  4.429e-02   1.720 0.085565 .
## device_brandXOLO       -2.568e-02  5.238e-02  -0.490 0.623985
## device_brandZTE        -6.374e-03  4.295e-02  -0.148 0.882031
## osiOS                  -1.395e-01  1.464e-01  -0.953 0.340690
## osOthers               -1.089e-01  3.060e-02  -3.561 0.000377 ***
## osWindows              -1.602e-02  4.104e-02  -0.390 0.696237
## screen_size             2.154e-02  3.100e-03   6.949 4.65e-12 ***
## X4gyes                  3.546e-02  1.512e-02   2.346 0.019073 *
## rear_camera_mp          2.115e-02  1.390e-03  15.216  < 2e-16 ***
```

```
## front_camera_mp         1.356e-02  1.094e-03  12.386  < 2e-16 ***
## ram                     2.054e-02  4.614e-03   4.452 8.86e-06 ***
## battery                -2.151e-05  7.421e-06  -2.899 0.003780 **
## weight                  1.087e-03  1.222e-04   8.892  < 2e-16 ***
## release_year            3.065e-02  4.361e-03   7.028 2.68e-12 ***
## days_used               4.775e-05  2.919e-05   1.636 0.102016
## normalized_new_price    4.362e-01  1.142e-02  38.185  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2325 on 2557 degrees of freedom
## Multiple R-squared:  0.8469, Adjusted R-squared:  0.8442
## F-statistic: 314.3 on 45 and 2557 DF,  p-value: < 2.2e-16
```

```
AIC(bcd_mdel)
```

```
## [1] -160.2183
```

```
BIC(bcd_mdel)
```

```
## [1] 115.4094
```

```
coef(bcd_mdel)
```

```
##           (Intercept)    device_brandAlcatel      device_brandApple
##         -6.055235e+01          -3.032950e-02           5.730179e-02
##       device_brandAsus device_brandBlackBerry     device_brandCelkon
##          5.109778e-02           1.696765e-01          -9.783200e-02
##    device_brandCoolpad      device_brandGionee     device_brandGoogle
##         -3.205464e-02           1.580418e-02          -4.317158e-02
##      device_brandHonor         device_brandHTC     device_brandHuawei
##          4.084860e-03          -1.786322e-02          -1.505475e-02
##    device_brandKarbonn        device_brandLava     device_brandLenovo
##         -2.751663e-02           9.271669e-03           2.282171e-02
##         device_brandLG       device_brandMeizu   device_brandMicromax
##         -4.533282e-02           2.996056e-03          -3.102629e-02
##  device_brandMicrosoft   device_brandMotorola      device_brandNokia
##          4.977796e-02          -3.427824e-02           8.860588e-02
##    device_brandOnePlus        device_brandOppo     device_brandOthers
##          6.289890e-03           5.916743e-03          -8.221584e-03
##  device_brandPanasonic      device_brandRealme    device_brandSamsung
##         -9.958189e-03           2.388479e-01          -3.130017e-02
##       device_brandSony       device_brandSpice       device_brandVivo
##         -5.757353e-02          -1.897179e-02          -3.377031e-02
##     device_brandXiaomi        device_brandXOLO        device_brandZTE
##          7.617136e-02          -2.568083e-02          -6.373548e-03
##                 osiOS               osOthers              osWindows
##         -1.394827e-01          -1.089367e-01          -1.602313e-02
##           screen_size                 X4gyes         rear_camera_mp
##          2.154112e-02           3.546300e-02           2.114567e-02
##        front_camera_mp                    ram                battery
##          1.355507e-02           2.054500e-02          -2.151235e-05
##                 weight           release_year              days_used
##          1.087018e-03           3.065151e-02           4.774726e-05
##   normalized_new_price
##          4.361640e-01
```

Answer:We choose model 2 or 4 because of its lower AIC and BIC values compared to model3 in model2 we choose device_brand, os, screeen_size, X4g, X5g, rear_camera_mp, front_camera_mp,ram,battery,weight,release_year,days_used,normalized_new_price note that there are more subcatogories of this as listed above in the intercept

b).(14 pts) Perform LASSO and RIDGE regression on the dataset “trainData” .Use `cv.glmnet()` to find the lambda value that minimizes the cross-validation error using 10 fold CV.

Answer the following questions for both models.

- State the value of the optimal lambda.
- Fit the model with 100 values for lambda.
- Extract coefficients at the optimal lambda. Which coefficients are selected? Compare the number of coefficients selected by both the models. Why are you seeing this behavior?
- Plot the coefficient path for both the models and compare.

```
See https://www.science.smith.edu/~jcrouser/SDS293/labs/lab10-r.html

Remember to use as.matrix() to process categorical predictors
```

```
x <- model.matrix(normalized_used_price ~ . - 1, data = trainData)  # Remove interc
ept
y <- trainData$normalized_used_price
cv.lasso <- cv.glmnet(x, y, alpha = 1, nfolds = 10)
cv.ridge <- cv.glmnet(x, y, alpha = 0, nfolds = 10)
lambda.lasso <- cv.lasso$lambda.min
lambda.lasso
```

```
## [1] 0.00470675
```

```
lambda.ridge <- cv.ridge$lambda.min
lambda.lasso
```

```
## [1] 0.00470675
```

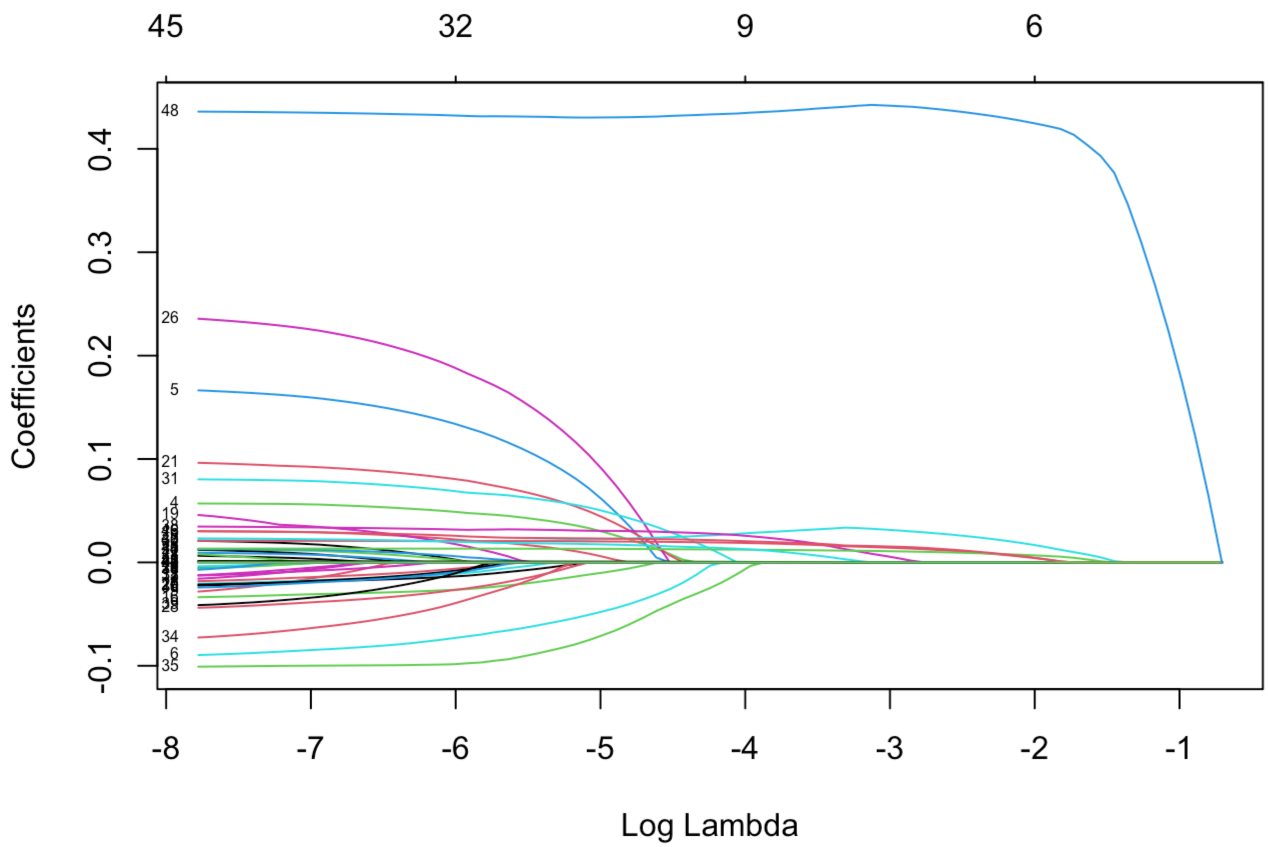
```
lasso.fit <- glmnet(x, y, alpha = 1, lambda = cv.lasso$lambda)
fit.ridge <- glmnet(x, y, alpha = 0, lambda = cv.ridge$lambda)
coeff.lasso <- coef(lasso.fit, s = lambda.lasso)
coeff.lasso
```

```
## 49 x 1 sparse Matrix of class "dgCMatrix"
##                                   s1
## (Intercept)           -4.616451e+01
## device_brandAcer        .
## device_brandAlcatel     .
## device_brandApple       .
## device_brandAsus        4.012075e-02
## device_brandBlackBerry  9.706403e-02
## device_brandCelkon     -5.889696e-02
## device_brandCoolpad     .
## device_brandGionee      .
## device_brandGoogle      .
## device_brandHonor       .
## device_brandHTC         .
## device_brandHuawei      .
## device_brandKarbonn     .
## device_brandLava        .
## device_brandLenovo      1.449733e-02
## device_brandLG         -1.748252e-02
## device_brandMeizu       .
## device_brandMicromax   -8.658737e-04
## device_brandMicrosoft   .
## device_brandMotorola    .
## device_brandNokia       6.230912e-02
## device_brandOnePlus     .
## device_brandOppo        .
## device_brandOthers      .
## device_brandPanasonic   .
## device_brandRealme      1.383147e-01
## device_brandSamsung    -5.012239e-03
## device_brandSony       -1.030376e-02
## device_brandSpice       .
## device_brandVivo        .
## device_brandXiaomi      5.984542e-02
## device_brandXOLO        .
## device_brandZTE         .
## osiOS                  -1.027299e-02
## osOthers               -8.565267e-02
## osWindows               .
## screen_size             2.124363e-02
## X4gyes                  3.144220e-02
## X5gyes                  .
## rear_camera_mp          2.047407e-02
## front_camera_mp         1.320783e-02
## internal_memory         .
## ram                     1.832269e-02
## battery                 .
## weight                  8.356267e-04
## release_year            2.353930e-02
## days_used               .
## normalized_new_price    4.308966e-01
```

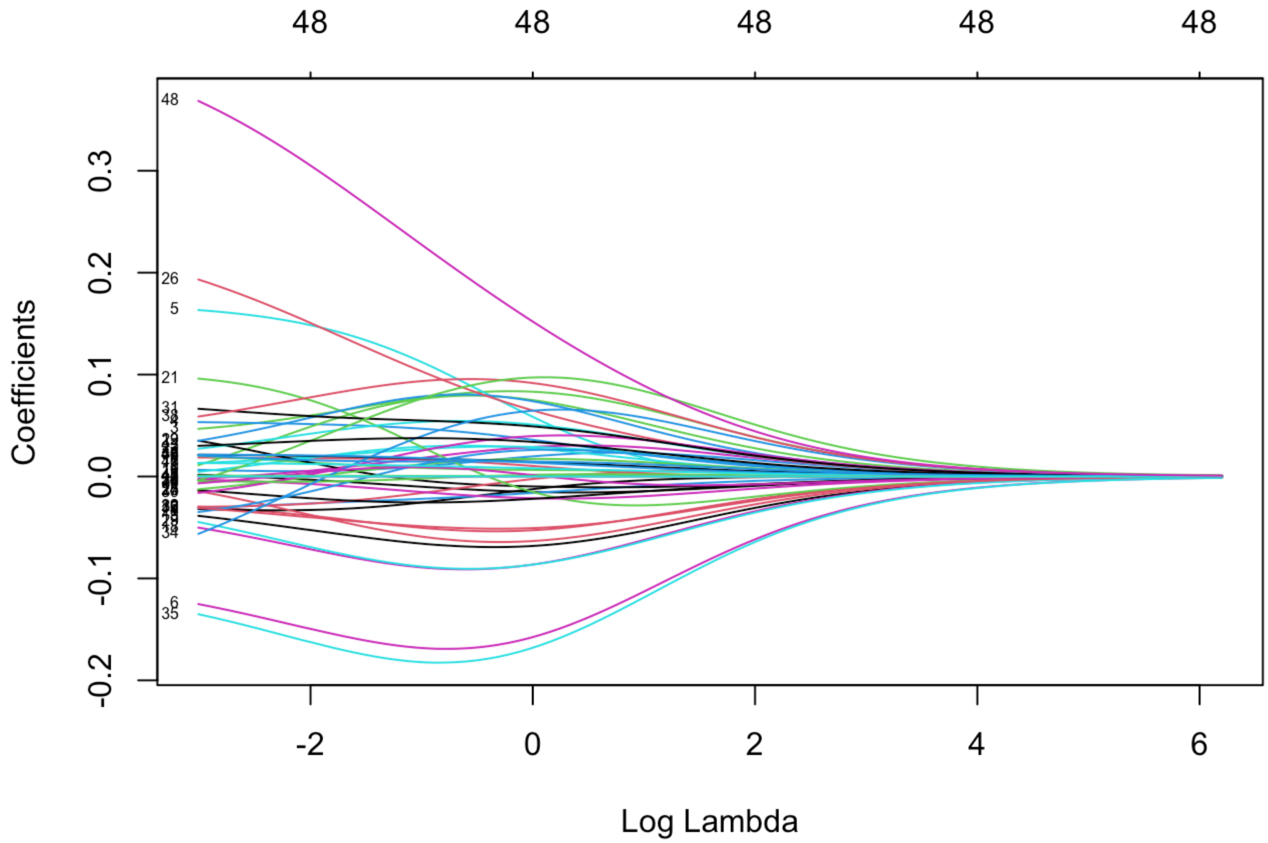
```
ridge.coeff <- coef(fit.ridge, s = lambda.ridge)
ridge.coeff
```

```
## 49 x 1 sparse Matrix of class "dgCMatrix"
##                                   s1
## (Intercept)            -4.005685e+01
## device_brandAcer        1.737830e-03
## device_brandAlcatel    -2.951509e-02
## device_brandApple       4.677269e-02
## device_brandAsus        5.337188e-02
## device_brandBlackBerry  1.633308e-01
## device_brandCelkon     -1.251803e-01
## device_brandCoolpad    -3.191263e-02
## device_brandGionee      1.866418e-02
## device_brandGoogle      1.109220e-02
## device_brandHonor       6.494559e-03
## device_brandHTC         4.607122e-03
## device_brandHuawei     -2.314148e-03
## device_brandKarbonn    -3.859656e-02
## device_brandLava       -1.434145e-02
## device_brandLenovo      2.034079e-02
## device_brandLG         -3.110637e-02
## device_brandMeizu       1.326207e-02
## device_brandMicromax   -5.014532e-02
## device_brandMicrosoft   3.508488e-02
## device_brandMotorola   -2.970566e-02
## device_brandNokia       9.603936e-02
## device_brandOnePlus     3.504609e-02
## device_brandOppo        2.721308e-02
## device_brandOthers     -3.206695e-03
## device_brandPanasonic  -1.349413e-02
## device_brandRealme      1.933051e-01
## device_brandSamsung    -1.253960e-02
## device_brandSony       -3.488407e-02
## device_brandSpice      -4.466596e-02
## device_brandVivo       -1.611249e-02
## device_brandXiaomi      6.646034e-02
## device_brandXOLO       -3.079675e-02
## device_brandZTE        -4.414064e-03
## osiOS                  -5.630535e-02
## osOthers               -1.351054e-01
## osWindows              -6.717087e-03
## screen_size             2.137024e-02
## X4gyes                  5.880012e-02
## X5gyes                 -3.751981e-03
## rear_camera_mp          2.134755e-02
## front_camera_mp         1.319329e-02
## internal_memory         1.166942e-04
## ram                     3.008852e-02
## battery                 4.991041e-06
## weight                  8.476385e-04
## release_year            2.061733e-02
## days_used               2.372635e-05
## normalized_new_price    3.685065e-01
```

```
plot(lasso.fit, xvar = "lambda", label = TRUE)
```




```
plot(fit.ridge, xvar = "lambda", label = TRUE)
```



Answer: The ideal lambda value of 0.00470675 was determined for both LASSO and Ridge regression models. When these models were trained with 100 different lambda values, LASSO tended to select fewer coefficients, assigning many to zero, such as Apple and Google brands, as well as attributes like internal memory, which were entirely excluded at the optimal lambda. In contrast, Ridge regression retained all coefficients, although their magnitudes were reduced. This contrast is evident in the coefficient paths: as lambda increases, LASSO's path shows many coefficients converging to zero, indicating its capacity for variable selection and model simplification. Meanwhile, Ridge regression's path demonstrates that all coefficients shrink towards zero without reaching it, thereby maintaining all predictors in the model.

c).(4 pts) Apply Principal Component Analysis and then create a regression model using the first few principal components, name it `pca_model`.

```
You can use the R function prcomp for PCA. (Note that to first scale the data, you
can include scale. = TRUE to scale as part of the PCA function. Don't forget that,
to make a prediction for the new city, you'll need to unscale the coefficients (i.
e., do the scaling calculation in reverse)!)
```

```
set.seed(100)
x <- trainData[, sapply(trainData, is.numeric)]  # selecting only numeric predictor
s
x_scaled <- scale(x)
pca <- prcomp(x_scaled, center = TRUE, scale. = TRUE)
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     2.1788 1.3871 1.1588 0.9613 0.86820 0.60602 0.57893
## Proportion of Variance 0.4315 0.1749 0.1221 0.0840 0.06853 0.03339 0.03047
## Cumulative Proportion  0.4315 0.6065 0.7286 0.8125 0.88107 0.91446 0.94493
##                            PC8     PC9    PC10    PC11
## Standard deviation     0.46302 0.42146 0.33637 0.31720
## Proportion of Variance 0.01949 0.01615 0.01029 0.00915
## Cumulative Proportion  0.96442 0.98057 0.99085 1.00000
```

```
pca_scores <- pca$x[, 1:3]  # Extract the scores of the first three principal compo
nents
pca_model <- lm(normalized_used_price ~ pca_scores, data = trainData)
summary(pca_model)
```

```
##
## Call:
## lm(formula = normalized_used_price ~ pca_scores, data = trainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.01305 -0.12090  0.00473  0.12656  1.27105
##
## Coefficients:
##                Estimate Std. Error  t value Pr(>|t|)
## (Intercept)    4.344162   0.003941 1102.399  < 2e-16 ***
## pca_scoresPC1 -0.241692   0.001809 -133.605  < 2e-16 ***
## pca_scoresPC2  0.014773   0.002841    5.199 2.16e-07 ***
## pca_scoresPC3 -0.146620   0.003401  -43.109  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2011 on 2599 degrees of freedom
## Multiple R-squared:  0.8836, Adjusted R-squared:  0.8835
## F-statistic:  6579 on 3 and 2599 DF,  p-value: < 2.2e-16
```

Part 4 [24 pts]: PREDICTION MODEL COMPARISON

a).(14 pts) Using the testData, use the following models to predict the normalized price of the devices:

- model1 (Part 2a)
- model2 (Part 3a)
- model3 (Part 3a)
- model4 (Part 3a)
- Lasso (Part 3b)

- Ridge (Part 3b)
- pca_model (Part 3c)

Show the first five predictions using each model along with their true values. Are the values different?

```
set.seed(100)
test_matrix <- model.matrix(normalized_used_price ~ . - 1, data = testData)

predictions_model1 <- predict(model1, newdata = testData)
predictions_model2 <- predict(subset_best_mdel, newdata = testData)
predictions_model3 <- predict(fcd_mdel, newdata = testData)
predictions_model4 <- predict(bcd_mdel, newdata = testData)


predictions_lasso <- predict(lasso.fit, s = lambda.lasso, newx = as.matrix(test_mat
rix))
predictions_ridge <- predict(fit.ridge, s = lambda.ridge, newx = as.matrix(test_mat
rix))


test_scores <- predict(pca, newdata = scale(testData[, sapply(testData, is.numeri
c)]))
predictions_pca <- predict(pca_model, newdata = data.frame(pca_scores = test_scores
[, 1:3]))
```

```
## Warning: 'newdata' had 650 rows but variables found have 2603 rows
```

```
testData$normalized_used_price[1:5]
```

```
## [1] 4.072440 5.361902 3.183870 3.764451 4.603969
```

```
head(predictions_model1, 5)
```

```
##      503     2035     3033     3052     2967
## 4.194178 5.179859 3.541844 3.508612 4.528066
```

```
head(predictions_model2, 5)
```

```
##      503     2035     3033     3052     2967
## 4.195302 5.175700 3.541426 3.504972 4.527867
```

```
head(predictions_model3, 5)
```

```
##      503     2035     3033     3052     2967
## 4.194178 5.179859 3.541844 3.508612 4.528066
```

```
head(predictions_model4, 5)
```

```
##      503     2035     3033     3052     2967
## 4.195302 5.175700 3.541426 3.504972 4.527867
```

```
head(predictions_lasso, 5)
```

```
##            s1
## 503  4.176842
## 2035 5.148509
## 3033 3.529444
## 3052 3.510980
## 2967 4.505115
```

```
head(predictions_ridge, 5)
```

```
##            s1
## 503  4.199597
## 2035 5.129050
## 3033 3.557035
## 3052 3.500378
## 2967 4.502729
```

```
head(predictions_pca, 5)
```

```
##        1        2        3        4        5
## 4.250572 5.221011 4.484000 4.551124 3.903306
```

Answer:The majority of the values appear to be fairly close to the actual value, despite some differences in values.

b). (10 pts) Compare the predictions using mean squared prediction error. Which model performed the best? Reasoning your answer and discussion your choice(s).

```
set.seed(100)
set.seed(100)

mspe_model1 <- mean((predictions_model1 - testData$normalized_used_price)^2)
mspe_model2 <- mean((predictions_model2 - testData$normalized_used_price)^2)
mspe_model3 <- mean((predictions_model3 - testData$normalized_used_price)^2)
mspe_model4 <- mean((predictions_model4 - testData$normalized_used_price)^2)
mspe_lasso <- mean((predictions_lasso - testData$normalized_used_price)^2)
mspe_ridge <- mean((predictions_ridge - testData$normalized_used_price)^2)
mspe_pca <- mean((predictions_pca - testData$normalized_used_price)^2)
```

```
## Warning in predictions_pca - testData$normalized_used_price: longer object
## length is not a multiple of shorter object length
```

```
cat("MSPE Model 1:", mspe_model1, "\n")
```

```
## MSPE Model 1: 0.05464308
```

```
cat("MSPE Model 2:", mspe_model2, "\n")
```

```
## MSPE Model 2: 0.05465481
```

```
cat("MSPE Model 3:", mspe_model3, "\n")
```

```
## MSPE Model 3: 0.05464308
```

```
cat("MSPE Model 4:", mspe_model4, "\n")
```

```
## MSPE Model 4: 0.05465481
```

```
cat("MSPE Lasso:", mspe_lasso, "\n")
```

```
## MSPE Lasso: 0.05399679
```

```
cat("MSPE Ridge:", mspe_ridge, "\n")
```

```
## MSPE Ridge: 0.05448073
```

```
cat("MSPE PCA Model:", mspe_pca, "\n")
```

```
## MSPE PCA Model: 0.5908957
```

Answer: The MSPE of all the models are as shown above 1.The Lasso model achieves the lowest MSPE of 0.05399679, indicating its superior performance compared to other models. Its capability to conduct variable selection and regularization effectively reduces overfitting and enhances prediction accuracy on the testData. 2.Following closely, both Model 1 and Model 3 exhibit identical MSPE values of 0.05464308, demonstrating reasonably accurate predictions, nearly matching the Lasso model's performance. 3.The Ridge model follows with a slightly higher MSPE of 0.05448073, still better than Models 2 and 4. 4.However, the PCA Model shows a significantly higher MSPE of 0.5908957, suggesting poorer prediction performance compared to other models. This discrepancy might arise from the loss of original data variability captured by the principal components, implying that crucial predictive information was not retained. 5. In summary, the Lasso model excels due to its regularization technique, effectively simplifying the model while maintaining or enhancing predictive accuracy, particularly beneficial in scenarios with many variables or high-dimensional datasets. Conversely, the PCA model's poor performance results from reducing the dataset to principal components, potentially discarding essential predictive information.