BA 810 Team Project

Yixuan Wang

2/19/2021

Mobile price classification

How to best predict the price range of a mobile phone based on technical features

Load useful libraries

```
library(data.table)
library(dplyr)
library(stringr)
library(caTools)
library(caret)
library(randomForest)
library(rpart)
library(rpart.plot)
```

Load Dataset

```
m <- fread("/Users/wangyixuan/Desktop/BA810 Supervised machine learning/data/processe
d_train.csv")
```

```
head(m, 1)
```

```
##    battery_power blue clock_speed dual_sim fc four_g int_memory m_dep mobile_wt
## 1:           842    0         2.2        0  1      0          7   0.6       188
##    n_cores pc px_height px_width  ram sc_h sc_w talk_time three_g touch_screen
## 1:       2  2        20      756 2549    9    7        19       0            0
##    wifi price_range price_binary p0 p1 p2 p3
## 1:    1           1            0  0  1  0  0
```

```
dim(m)
```

```
## [1] 2000   26
```

1. Binary

```
m_binary <- m[, !c("price_range", "p0", "p1", "p2", "p3")]
```

Split our data set on training and testing subset.

```
set.seed(810)
sampleSplit <- sample.split(Y=m_binary$price_binary, SplitRatio=0.7)
trainSet <- subset(x=m_binary, sampleSplit==TRUE)
testSet <- subset(x=m_binary, sampleSplit==FALSE)
```

1.1 Logistic regression

```
log_model_binary <- glm(price_binary ~ ., family=binomial(link='logit'), data=trainSe
t)
```

```
summary(log_model_binary)
```

```
##
## Call:
## glm(formula = price_binary ~ ., family = binomial(link = "logit"),
##     data = trainSet)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.592   0.000   0.000   0.000   2.969
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -3.107e+02  1.030e+02  -3.018 0.002548 **
## battery_power  5.582e-02  1.890e-02   2.953 0.003149 **
## blue          -4.575e-01  2.111e+00  -0.217 0.828441
## clock_speed    9.472e-01  9.297e-01   1.019 0.308269
## dual_sim      -8.063e-01  1.425e+00  -0.566 0.571616
## fc            -7.492e-02  2.968e-01  -0.252 0.800702
## four_g        -2.352e+00  1.830e+00  -1.285 0.198899
## int_memory     1.369e-01  6.131e-02   2.234 0.025515 *
## m_dep         -4.170e+00  2.467e+00  -1.690 0.090976 .
## mobile_wt     -9.513e-02  3.353e-02  -2.838 0.004547 **
## n_cores        3.221e-01  3.063e-01   1.052 0.292982
## pc             2.103e-01  1.553e-01   1.354 0.175625
## px_height      3.019e-02  9.071e-03   3.328 0.000873 ***
## px_width       3.391e-02  1.219e-02   2.781 0.005422 **
## ram            8.798e-02  2.893e-02   3.041 0.002358 **
## sc_h          -1.976e-02  2.029e-01  -0.097 0.922421
## sc_w           2.505e-01  2.108e-01   1.188 0.234763
## talk_time      6.372e-02  1.186e-01   0.537 0.591108
## three_g        2.473e+00  1.814e+00   1.363 0.172915
## touch_screen  -8.278e-01  1.540e+00  -0.538 0.590831
## wifi          -3.303e+00  1.882e+00  -1.755 0.079244 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1940.812  on 1399  degrees of freedom
## Residual deviance:   32.228  on 1379  degrees of freedom
## AIC: 74.228
##
## Number of Fisher Scoring iterations: 15
```

```
probabs <- predict(log_model_binary, testSet[,!c("price_binary")],type='response')
preds <- ifelse(probabs > 0.5, 1, 0)
```

```
confusionMatrix(factor(preds), factor(testSet$price_binary))
```

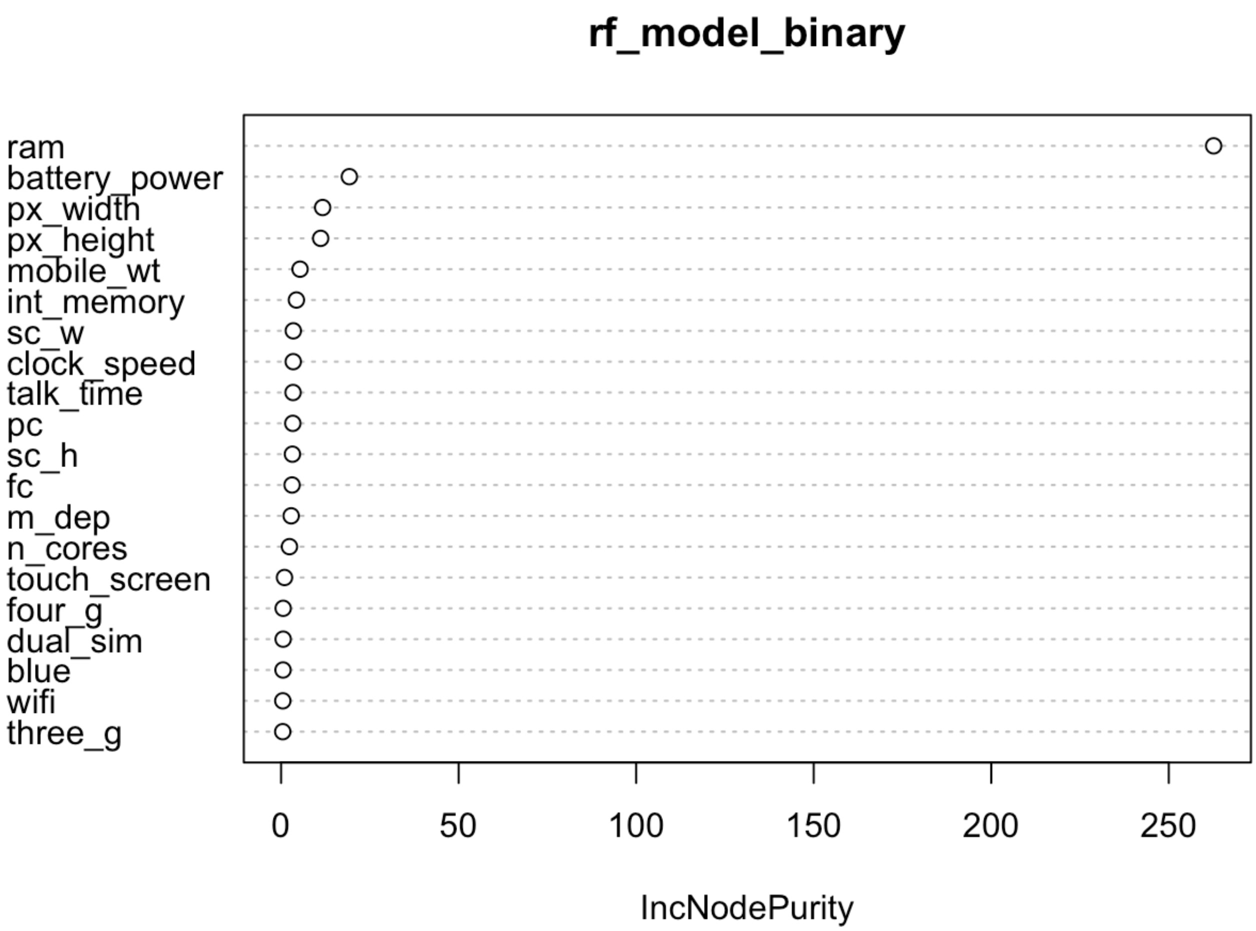
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 297   2
##          1   3 298
##
##                Accuracy : 0.9917
##                  95% CI : (0.9807, 0.9973)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9833
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9900
##             Specificity : 0.9933
##          Pos Pred Value : 0.9933
##          Neg Pred Value : 0.9900
##              Prevalence : 0.5000
##          Detection Rate : 0.4950
##    Detection Prevalence : 0.4983
##       Balanced Accuracy : 0.9917
##
##        'Positive' Class : 0
##
```

Overall, our logistic regression model is correct in roughly 99.17% of the test cases.

1.2 Random Forest

```
rf_model_binary <- randomForest(
  price_binary ~ .,
  data=trainSet
)
```

```
varImpPlot(rf_model_binary)
```




```
probabs <- predict(rf_model_binary, testSet[,!c("price_binary")])
preds <- ifelse(probabs > 0.5, 1, 0)
```

```
confusionMatrix(factor(preds), factor(testSet$price_binary))
```

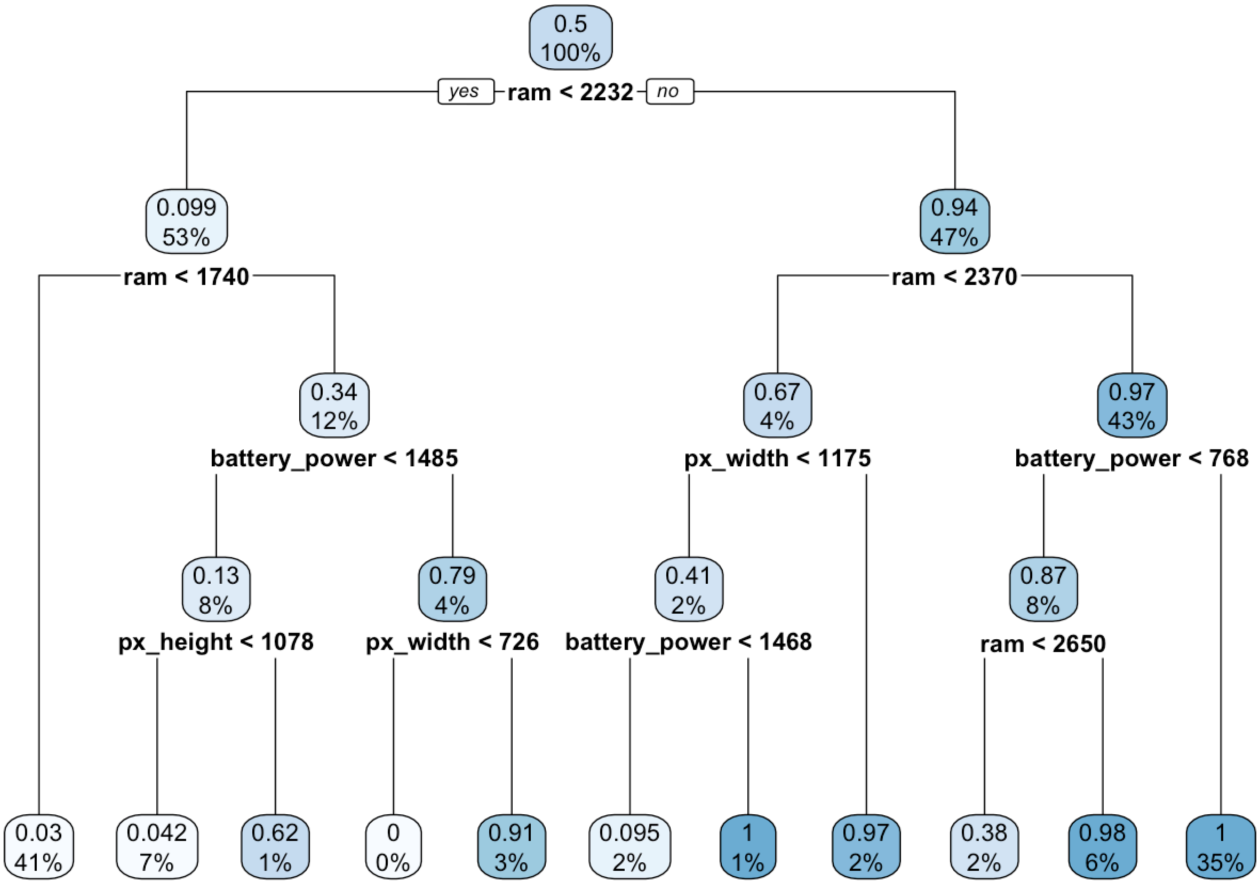
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 279  12
##          1  21 288
##
##                Accuracy : 0.945
##                  95% CI : (0.9236, 0.9618)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.89
##
##  Mcnemar's Test P-Value : 0.1637
##
##             Sensitivity : 0.9300
##             Specificity : 0.9600
##          Pos Pred Value : 0.9588
##          Neg Pred Value : 0.9320
##              Prevalence : 0.5000
##          Detection Rate : 0.4650
##    Detection Prevalence : 0.4850
##       Balanced Accuracy : 0.9450
##
##        'Positive' Class : 0
##
```

Overall, our random forest model is correct in roughly 94.5% of the test cases.

1.3 Decision Tree

```
dt_model_binary <- rpart(
  price_binary ~ .,
  data=trainSet,
  control = rpart.control(cp = 0.01)
)
```

```
rpart.plot(dt_model_binary)
```




```
probabs <- predict(dt_model_binary, testSet[,!c("price_binary")])
preds <- ifelse(probabs > 0.5, 1, 0)
```

```
confusionMatrix(table(testSet$price_binary, preds))
```

```
## Confusion Matrix and Statistics
##
##    preds
##       0   1
##   0 281  19
##   1  24 276
##
##                Accuracy : 0.9283
##                  95% CI : (0.9047, 0.9477)
##     No Information Rate : 0.5083
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8567
##
##  Mcnemar's Test P-Value : 0.5419
##
##             Sensitivity : 0.9213
##             Specificity : 0.9356
##          Pos Pred Value : 0.9367
##          Neg Pred Value : 0.9200
##              Prevalence : 0.5083
##          Detection Rate : 0.4683
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.9285
##
##        'Positive' Class : 0
##
```

Overall, our logistic regression model is correct in roughly 92.83% of the test cases.

2. Multiple Binary

```
m_multi <- m[, !c("price_binary")]
```

2.1 Logistic regression

```
set.seed(810)

sampleSplit_multi <- sample.split(Y=m_multi$price_range, SplitRatio=0.7)
trainSet_multi <- subset(x=m_multi, sampleSplit_multi==TRUE)
testSet_multi <- subset(x=m_multi, sampleSplit_multi==FALSE)
```

```
log_model_0 <- glm(p0 ~ ., family=binomial(link='logit'),
                   data=trainSet_multi[,!c("price_range", "p1", "p2", "p3")])
log_model_1 <- glm(p1 ~ ., family=binomial(link='logit'),
                   data=trainSet_multi[,!c("price_range","p0", "p2", "p3")])
log_model_2 <- glm(p2 ~ ., family=binomial(link='logit'),
                   data=trainSet_multi[,!c("price_range", "p0", "p1", "p3")])
log_model_3 <- glm(p3 ~ ., family=binomial(link='logit'),
                   data=trainSet_multi[,!c("price_range", "p0", "p1", "p2")])
```

```
pr0_log <- predict(log_model_0, testSet_multi[,!c("price_range","p0", "p1", "p2", "p3
")],
               type='response')

pr1_log <- predict(log_model_1, testSet_multi[,!c("price_range","p0", "p1", "p2", "p3
")],
               type='response')

pr2_log <- predict(log_model_2, testSet_multi[,!c("price_range","p0", "p1", "p2", "p3
")],
               type='response')

pr3_log <- predict(log_model_3, testSet_multi[,!c("price_range","p0", "p1", "p2", "p3
")],
               type='response')

res_log <- cbind(pr0_log,pr1_log,pr2_log,pr3_log)
label_log <- apply(res_log,1,which.max)-1
```

```
table(label_log,testSet_multi$price_range)
```

```
##
## label_log   0   1   2   3
##         0 142   4   0   0
##         1   5 113  31   0
##         2   3  33 116   3
##         3   0   0   3 147
```

```
accuracy_multi_log <- sum(label_log==testSet_multi$price_range)/length(label_log)
accuracy_multi_log
```

```
## [1] 0.8633333
```

2.2 Random forest

```
set.seed(810)

sampleSplit_multi <- sample.split(Y=m_multi$price_range, SplitRatio=0.7)
trainSet_multi <- subset(x=m_multi, sampleSplit_multi==TRUE)
testSet_multi <- subset(x=m_multi, sampleSplit_multi==FALSE)
```

```
rf_model_0 <- randomForest(p0 ~ .,
                           data=trainSet_multi[,!c("price_range","p1", "p2", "p3")])
rf_model_1 <- randomForest(p1 ~ .,
                           data=trainSet_multi[,!c("price_range","p0", "p2", "p3")])
rf_model_2 <- randomForest(p2 ~ .,
                           data=trainSet_multi[,!c("price_range","p0", "p1", "p3")])
rf_model_3 <- randomForest(p3 ~ .,
                           data=trainSet_multi[,!c("price_range","p0", "p1", "p2")])
```

```
pr0_rf <- predict(rf_model_0, testSet_multi[,!c("price_range","p0", "p1", "p2", "p3")
])

pr1_rf <- predict(rf_model_1, testSet_multi[,!c("price_range","p0", "p1", "p2", "p3")
])

pr2_rf <- predict(rf_model_2, testSet_multi[,!c("price_range","p0", "p1", "p2", "p3")
])

pr3_rf <- predict(rf_model_3, testSet_multi[,!c("price_range","p0", "p1", "p2", "p3")
])

res_rf <- cbind(pr0_rf,pr1_rf,pr2_rf,pr3_rf)
label_rf <- apply(res_rf,1,which.max)-1
```

```
table(label_rf,testSet_multi$price_range)
```

```
##
## label_rf   0   1   2   3
##        0 138  16   0   0
##        1  12 121  18   0
##        2   0  13 116  15
##        3   0   0  16 135
```

```
accuracy_multi_rf <- sum(label_rf==testSet_multi$price_range)/length(label_rf)
accuracy_multi_rf
```

```
## [1] 0.85
```

2.3 Decision Tree

```
set.seed(810)

sampleSplit_multi <- sample.split(Y=m_multi$price_range, SplitRatio=0.7)
trainSet_multi <- subset(x=m_multi, sampleSplit_multi==TRUE)
testSet_multi <- subset(x=m_multi, sampleSplit_multi==FALSE)
```

```
dt_model_0 <- rpart(p0 ~ ., control = rpart.control(cp = 0.01),
                    data=trainSet_multi[,!c("price_range", "p1", "p2", "p3")])
dt_model_1 <- rpart(p1 ~ ., control = rpart.control(cp = 0.01),
                    data=trainSet_multi[,!c("price_range","p0", "p2", "p3")])
dt_model_2 <- rpart(p2 ~ ., control = rpart.control(cp = 0.01),
                    data=trainSet_multi[,!c("price_range", "p0", "p1", "p3")])
dt_model_3 <- rpart(p3 ~ ., control = rpart.control(cp = 0.01),
                    data=trainSet_multi[,!c("price_range", "p0", "p1", "p2")])
```

```
pr0_dt <- predict(dt_model_0, testSet_multi[,!c("price_range","p0", "p1", "p2", "p3")
])

pr1_dt <- predict(dt_model_1, testSet_multi[,!c("price_range","p0", "p1", "p2", "p3")
])

pr2_dt <- predict(dt_model_2, testSet_multi[,!c("price_range","p0", "p1", "p2", "p3")
])

pr3_dt <- predict(dt_model_3, testSet_multi[,!c("price_range","p0", "p1", "p2", "p3")
])

res_dt <- cbind(pr0_dt,pr1_dt,pr2_dt,pr3_dt)
label_dt <- apply(res_dt,1,which.max)-1
```

```
table(label_dt,testSet_multi$price_range)
```

```
##
## label_dt   0   1   2   3
##        0 134  20   0   0
##        1  16 115  17   0
##        2   0  13 114  17
##        3   0   2  19 133
```

```
accuracy_multi_dt <- sum(label_dt==testSet_multi$price_range)/length(label_dt)
accuracy_multi_dt
```

```
## [1] 0.8266667
```

3.Summary

```
accuracy_summary <- matrix(c(0.9917, 0.945, 0.9283, 0.8633, 0.85, 0.8267),ncol=2)
colnames(accuracy_summary) <- c("accuracy_binary", "accuracy_multi_binary")
rownames(accuracy_summary) <- c("logistic regreesion","random forest","decision tree"
)
accuracy_summary <- as.table(accuracy_summary)
accuracy_summary
```

```
##                     accuracy_binary accuracy_multi_binary
## logistic regreesion          0.9917                0.8633
## random forest                0.9450                0.8500
## decision tree                0.9283                0.8267
```