mag_final

Bingyu Sun

5/8/2019

##Data import & cleaning * Reclassify response variable to make it binary

```
apple = read_csv("./data/AppleStore.csv") %>%
  janitor::clean_names() %>%
  dplyr::select(-c(x1, id, track_name, currency, ver)) %>%
  mutate(size_bytes = round(size_bytes * 1e-6),
         cont_rating = factor(cont_rating, levels = c("4+", "9+", "12+", "17+")), #ascending order
         user_rating = ifelse(user_rating >= 4, "high", "low"),
         user_rating = factor(user_rating, levels = c("low", "high"))) %>%
  rename(size_megabytes = size_bytes) %>%
  filter(rating_count_tot != 0,
         user_rating_ver != 0) %>% #Remove apps with no user rating, and apps with no rating on current
  mutate(prime_genre = as.integer(ifelse(prime_genre == "Games", 1, 0))) %>%
  dplyr::select(-rating_count_tot, -rating_count_ver, -vpp_lic) %>%
  dplyr::select(user_rating, everything())
```

```
## Warning: Missing column names filled in: 'X1' [1]

## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   id = col_double(),
##   track_name = col_character(),
##   size_bytes = col_double(),
##   currency = col_character(),
##   price = col_double(),
##   rating_count_tot = col_double(),
##   rating_count_ver = col_double(),
##   user_rating = col_double(),
##   user_rating_ver = col_double(),
##   ver = col_character(),
##   cont_rating = col_character(),
##   prime_genre = col_character(),
##   sup_devices.num = col_double(),
##   ipadSc_urls.num = col_double(),
##   lang.num = col_double(),
##   vpp_lic = col_double()
## )
```

```
str(apple)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 5754 obs. of  9 variables:
##  $ user_rating     : Factor w/ 2 levels "low","high": 2 2 1 2 2 2 2 2 2 2 ...
##  $ size_megabytes  : num  101 159 101 129 93 10 228 130 49 70 ...
##  $ price           : num  3.99 0 0 0 0 0.99 0 0 9.99 3.99 ...
##  $ user_rating_ver : num  4.5 3.5 4.5 4.5 5 4 4.5 4.5 5 4 ...
##  $ cont_rating     : Factor w/ 4 levels "4+","9+","12+",..: 1 1 1 3 1 1 1 3 1 1 ...
##  $ prime_genre     : int  1 0 0 0 0 1 0 0 0 1 ...
##  $ sup_devices_num : num  38 37 37 37 37 47 37 37 37 38 ...
```

```
##  $ ipad_sc_urls_num: num  5 5 5 5 5 5 0 4 5 0 ...
##  $ lang_num        : num  10 23 3 9 45 1 19 1 1 10 ...
```

```r
table(apple$user_rating)
```

```
##
##  low high
## 1285 4469
```

##Split to train/test sets

```r
set.seed(1234)
#Split data to traning and testing
trRows = createDataPartition(apple$user_rating,
                             p = .75,
                             list = FALSE)
train_data = apple[trRows,]
test_data = apple[-trRows,]
#in matrix form
x_train = model.matrix(user_rating~., train_data)[,-1]
y_train = train_data$user_rating
x_test = model.matrix(user_rating~., test_data)[,-1]
y_test = test_data$user_rating

#CV method
ctrl1 = trainControl(method = "cv", number = 10)
ctrl2 = trainControl(method = "cv",
                     number = 10,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)
```

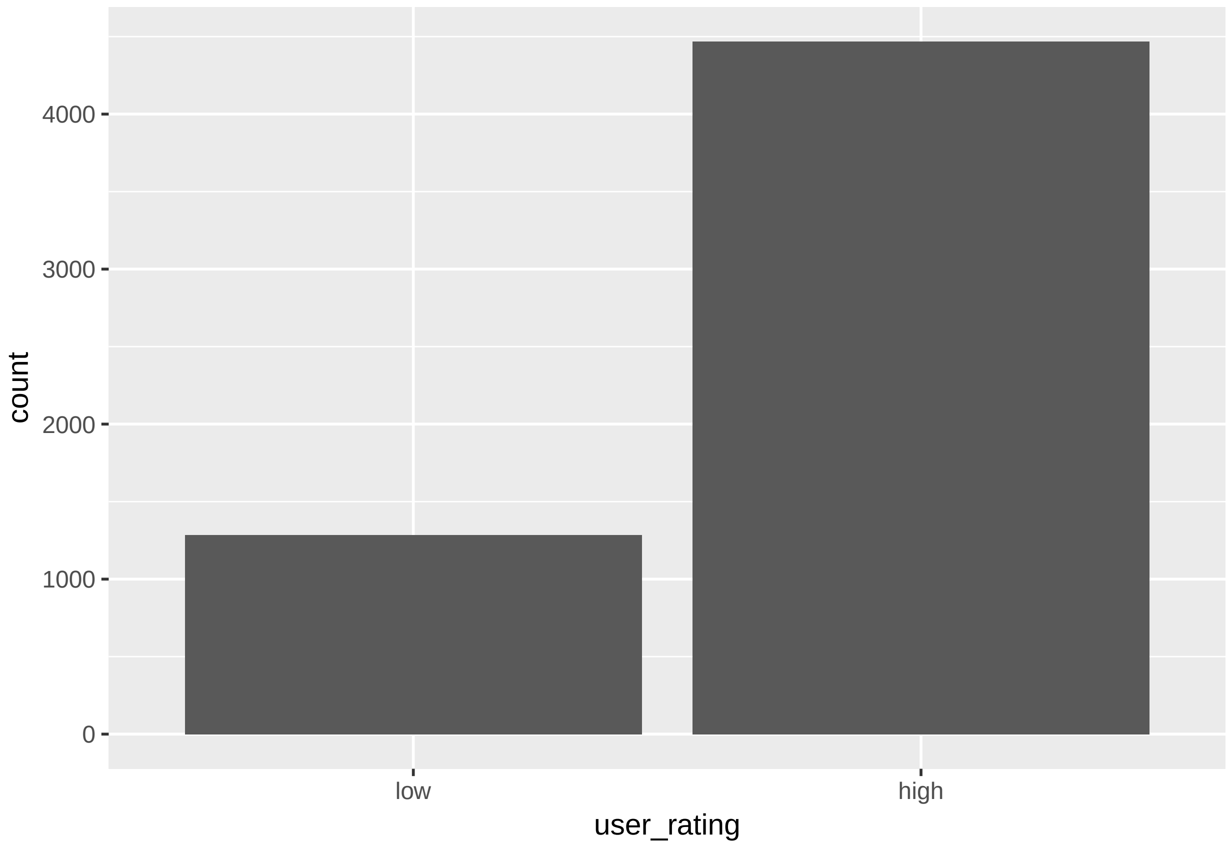
#Supervised learning

##Classification

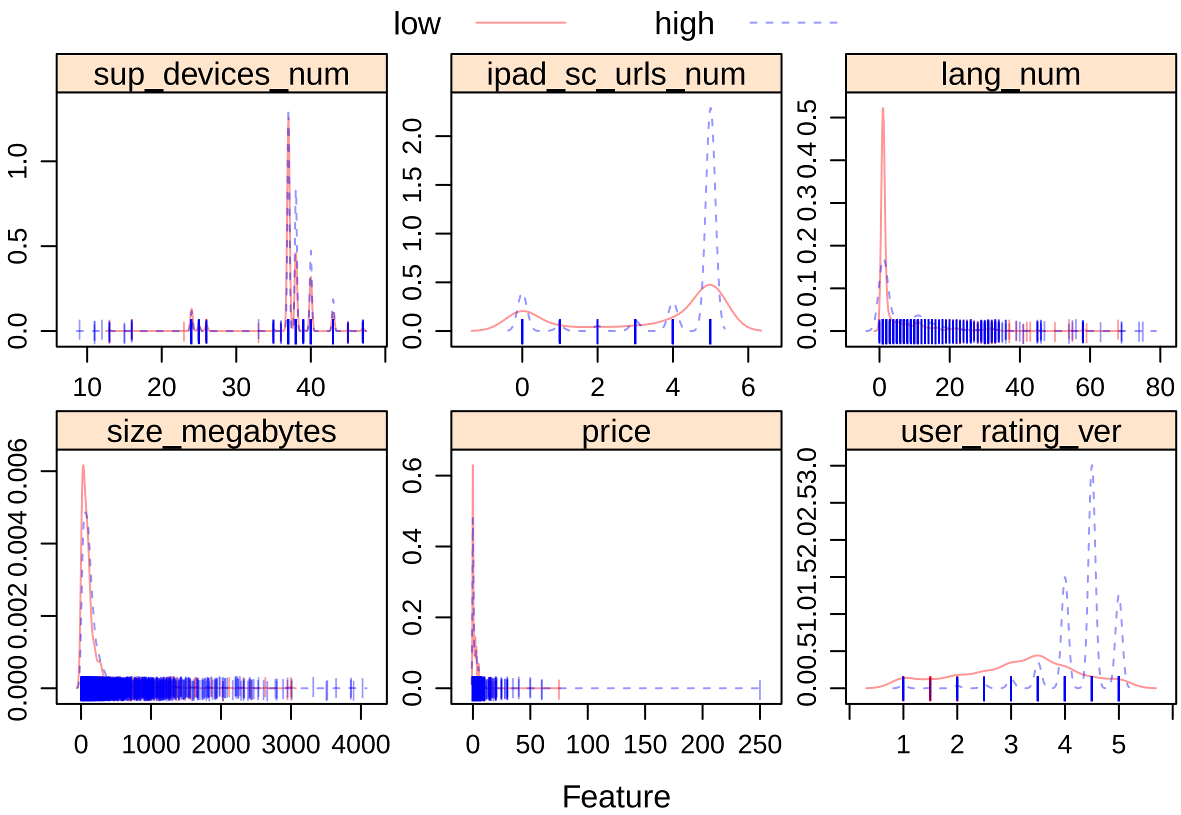
###For linear/non-linear decision boundary

####EDA

```r
#barplot for response
apple %>%
  ggplot(aes(x = user_rating)) +
  geom_bar()
```




```r
transparentTheme(trans = .4)
featurePlot(x = apple[, c(2:4, 7:9)],
            y = apple$user_rating,
            scales = list(x = list(relation = "free"),
                          y = list(relation = "free")),
            plot = "density", pch = "|",
            auto.key = list(columns = 2))
```



1a. Logistic Regression

- For large p, do penalization (ridge, lasso, elastic net)

```
glm.fit <- glm(user_rating~.,
               data = train_data,
               family = binomial)

contrasts(train_data$user_rating)
```

```
##      high
## low     0
## high    1
```

```
summary(glm.fit)
```

```
##
## Call:
## glm(formula = user_rating ~ ., family = binomial, data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9099   0.2445   0.3806   0.5370   2.9511
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -6.9282844  0.4912665 -14.103  < 2e-16 ***
## size_megabytes   -0.0001257  0.0001328  -0.947 0.343777
## price             0.0169367  0.0141117   1.200 0.230064
```

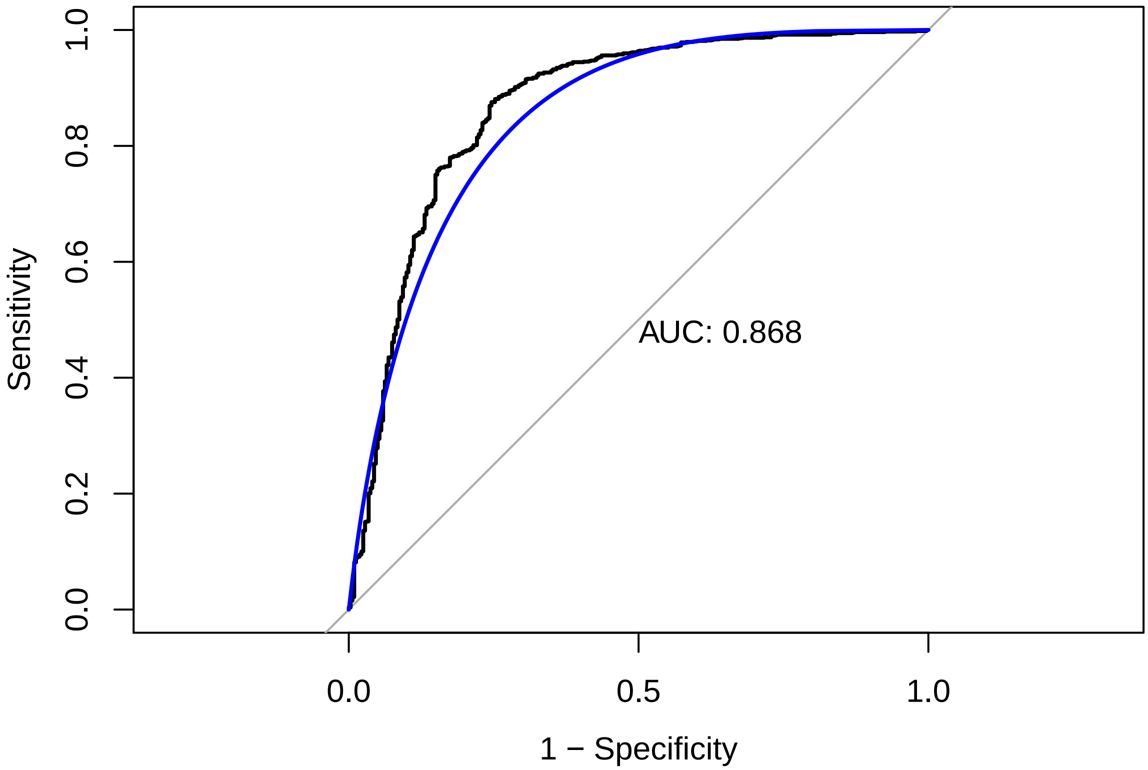
```
## user_rating_ver   1.6750654  0.0630475  26.568  < 2e-16 ***
## cont_rating9+     0.0893751  0.1447284   0.618 0.536881
## cont_rating12+    0.1454597  0.1328055   1.095 0.273393
## cont_rating17+   -0.1020242  0.1869774  -0.546 0.585306
## prime_genre       0.5241544  0.1072396   4.888 1.02e-06 ***
## sup_devices_num   0.0261461  0.0111411   2.347 0.018935 *
## ipad_sc_urls_num  0.0746664  0.0251981   2.963 0.003045 **
## lang_num          0.0198464  0.0059875   3.315 0.000918 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4584.6  on 4315  degrees of freedom
## Residual deviance: 3139.5  on 4305  degrees of freedom
## AIC: 3161.5
##
## Number of Fisher Scoring iterations: 5
```

```r
test.pred.prob  <- predict(glm.fit, newdata = test_data,
                           type = "response")
test.pred <- rep("low", length(test.pred.prob))
test.pred[test.pred.prob > 0.5] <- "high" #Bayes classifier (cutoff 0.5)

#Evaluate performance on the test data
confusionMatrix(data = factor(test.pred, levels = c("low", "high")),
                reference = test_data$user_rating,
                positive = "high")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  low high
##       low   149   35
##       high  172 1082
##
##                Accuracy : 0.8561
##                  95% CI : (0.8368, 0.8738)
##     No Information Rate : 0.7768
##     P-Value [Acc > NIR] : 2.187e-14
##
##                   Kappa : 0.5105
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9687
##             Specificity : 0.4642
##          Pos Pred Value : 0.8628
##          Neg Pred Value : 0.8098
##              Prevalence : 0.7768
##          Detection Rate : 0.7524
##    Detection Prevalence : 0.8720
##       Balanced Accuracy : 0.7164
##
##        'Positive' Class : high
```

```
##
```

```r
#Plot the test ROC curve
roc.glm <- roc(test_data$user_rating, test.pred.prob)
plot(roc.glm, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.glm), col = 4, add = TRUE)
```



For comparison, fit logistic regression using caret

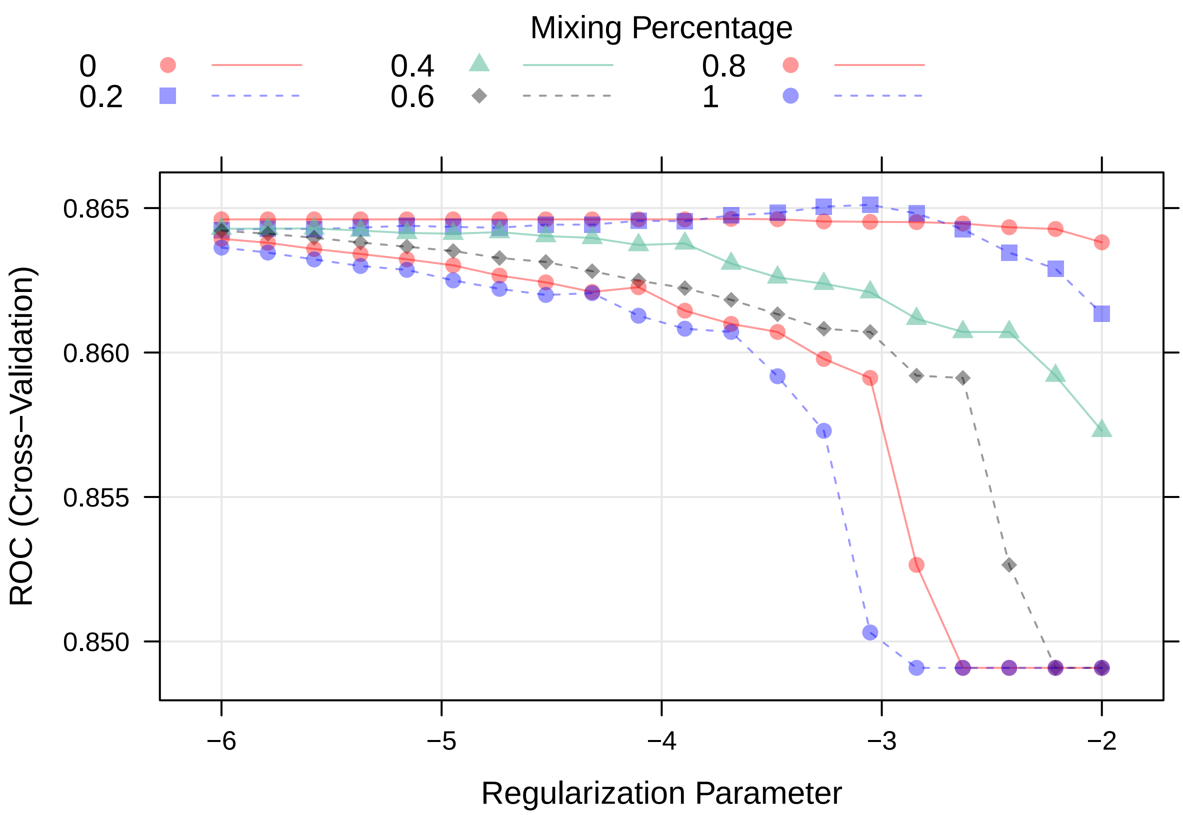
```r
set.seed(1234)
model.glm <- train(x = train_data[2:9],
                   y = train_data$user_rating,
                   method = "glm",
                   metric = "ROC",
                   trControl = ctrl2)
```

```
## Warning: Setting row names on a tibble is deprecated.

## Warning: Setting row names on a tibble is deprecated.

## Warning: Setting row names on a tibble is deprecated.

## Warning: Setting row names on a tibble is deprecated.

## Warning: Setting row names on a tibble is deprecated.

## Warning: Setting row names on a tibble is deprecated.

## Warning: Setting row names on a tibble is deprecated.

## Warning: Setting row names on a tibble is deprecated.
```

```
## Warning: Setting row names on a tibble is deprecated.

## Warning: Setting row names on a tibble is deprecated.

## Warning: Setting row names on a tibble is deprecated.
```

Consider penalization, do regularized logistic regression with glmnet, select the optimal tuning parameters

```
glmnGrid <- expand.grid(.alpha = seq(0, 1, length = 6),
                        .lambda = exp(seq(-6, -2, length = 20)))
set.seed(1234)
model.glmn <- train(x = x_train,
                    y = y_train,
                    method = "glmnet",
                    tuneGrid = glmnGrid,
                    metric = "ROC",
                    trControl = ctrl2)

plot(model.glmn, xTrans = function(x) log(x))
```



1b. GAM: consider non-linear covariates

2a. Linear discriminate analysis (LDA)

-Problem for logistic regression: if two classes are widely separated, model is unstable, large variance -Adv: So consider discriminant alaysis, for more than 2 classes, low-dimension views (good when have large p) * assume X normally distributed within each class, assume covariance are the same across classes

2b. Quadratic Discriminate analysis (QDA)

- No equal covariance assumption

3. Naivew Bayes

- good for large p, works for mixed p (continuous, categorical)

4. KNN

- center and scale first if method is based on distance
- super flexible -Disadv: no assumed model form, don't know relationship btw response and predictor

Tree-based methods

- No assumption, less strictive than linear methods, less flexible than knn
- Good interpretation

Regression

1. Regression tree

Classification

2. Classification tree