Homework 04 - BIOS 611

Michael Jetsupphasuk

10/10/2020

```
library(gbm)
library(gridExtra)
library(tidyverse)

dat = read_csv('500_Person_Gender_Height_Weight_Index.csv')
```

Problem 1

```
# create indicator for outcome variable gender
dat = dat %>% mutate(Female = as.numeric(Gender=='Female'))

# fit logistic regression model
logit_fit = glm(factor(Female) ~ Height + Weight, family = binomial(link='logit'), data = dat)

# report accuracy
logit_predict = as.numeric(predict(logit_fit, type='response') >= 0.5)
conf_mat_logit = table(logit_predict, dat$Female, dnn = c('predicted', 'observed'))
conf_mat_logit_prop = conf_mat_logit %>% prop.table()
accuracy_logit = conf_mat_logit_prop[1,1] + conf_mat_logit_prop[2,2]

# print out confusion matrix
conf_mat_logit_prop
```

```
##           observed
## predicted     0     1
##         0 0.094 0.084
##         1 0.396 0.426
```

The accuracy from a logistic regression with `Female` as the outcome variable and `Height` and `Weight` as the predictors yields an accuracy of 0.52. This accuracy is practically the same as the naive accuracy of predicting all female (0.51). See the confusion matrix above for a breakdown of prediction errors and accuracies by class.

Problem 2

```
# fit gbm
set.seed(789023511)
gbm_fit = gbm(Female ~ Height + Weight, data = dat, distribution = 'bernoulli')

# report accuracy
gbm_predict = as.numeric(predict(gbm_fit, type='response') >= 0.5)
conf_mat_gbm = table(gbm_predict, dat$Female, dnn = c('predicted', 'observed'))
conf_mat_gbm_prop = conf_mat_gbm %>% prop.table()
```

```
accuracy_gbm = conf_mat_gbm_prop[1,1] + conf_mat_gbm_prop[2,2]

# print out confusion matrix
conf_mat_gbm_prop
```

```
##           observed
## predicted     0     1
##         0 0.236 0.116
##         1 0.254 0.394
```

Using the same outcome and predictors, a gradient-boosting machine gives an accuracy of 0.63 using the default hyperparameters. The accuracy here is slightly better than the naive and logistic regression models. See the confusion matrix above for a breakdown of prediction errors and accuracies by class.

Problem 3

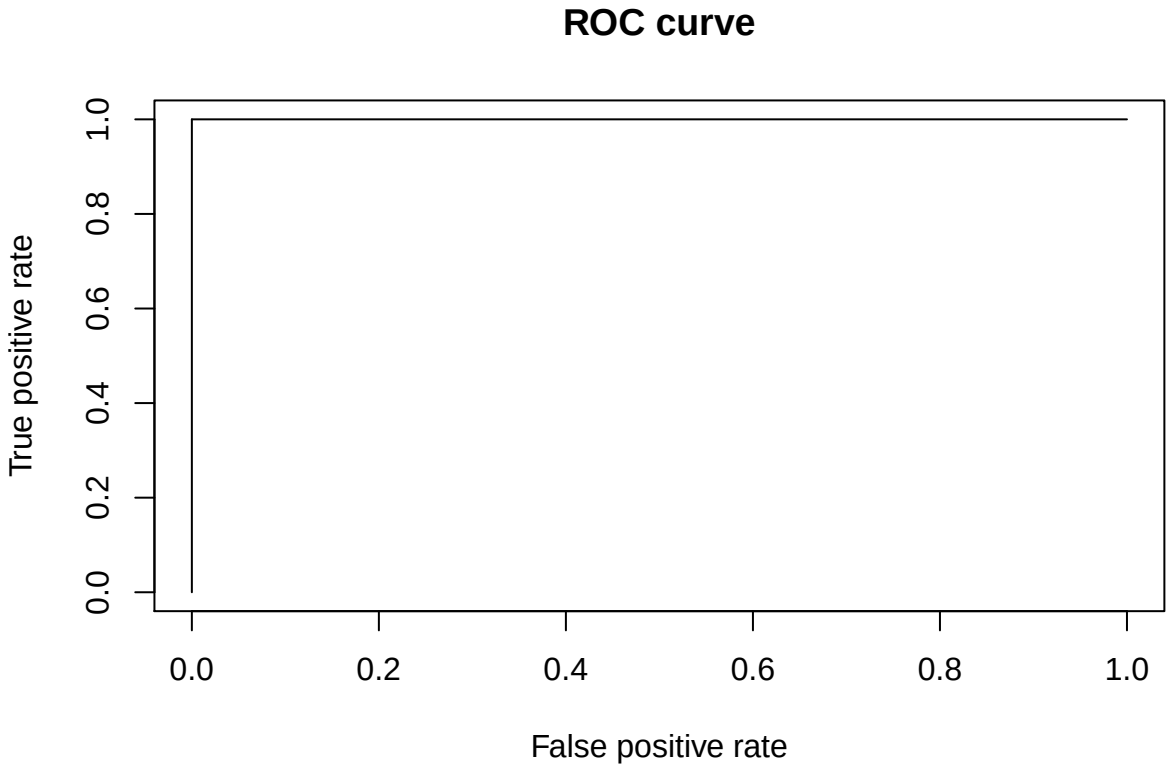
```
# fit gbm
set.seed(112852199)
dat2 = dat %>% filter(Female==0) %>% sample_n(50)
gbm_fit2 = gbm(Female ~ Height + Weight, data = dat2, distribution = 'bernoulli')

# get predictions (do not classify into male/female)
gbm_predict2 = predict(gbm_fit2, type='response')

# verify all predictions are male with perfect probability
all(gbm_predict2==0)
```

```
## [1] TRUE
```

Defining a new dataset of 50 random males, a GBM model correctly predicts all observations to be male. Therefore, the F1 score is 0 if we regard "males" to be negatives as before (recall that the F1 score is a ratio with true positives in the numerator). If we flip it and regard "males" to be positives then the precision and recall are 1 and so the F1 score is 1.

Problem 4

```
plot(x = c(0,1), y = c(1,1), type = 'l', ylim = c(0,1),
     xlab = 'False positive rate', ylab = 'True positive rate', main = 'ROC curve')
points(c(0,0), c(0,1), type  = 'l')
```



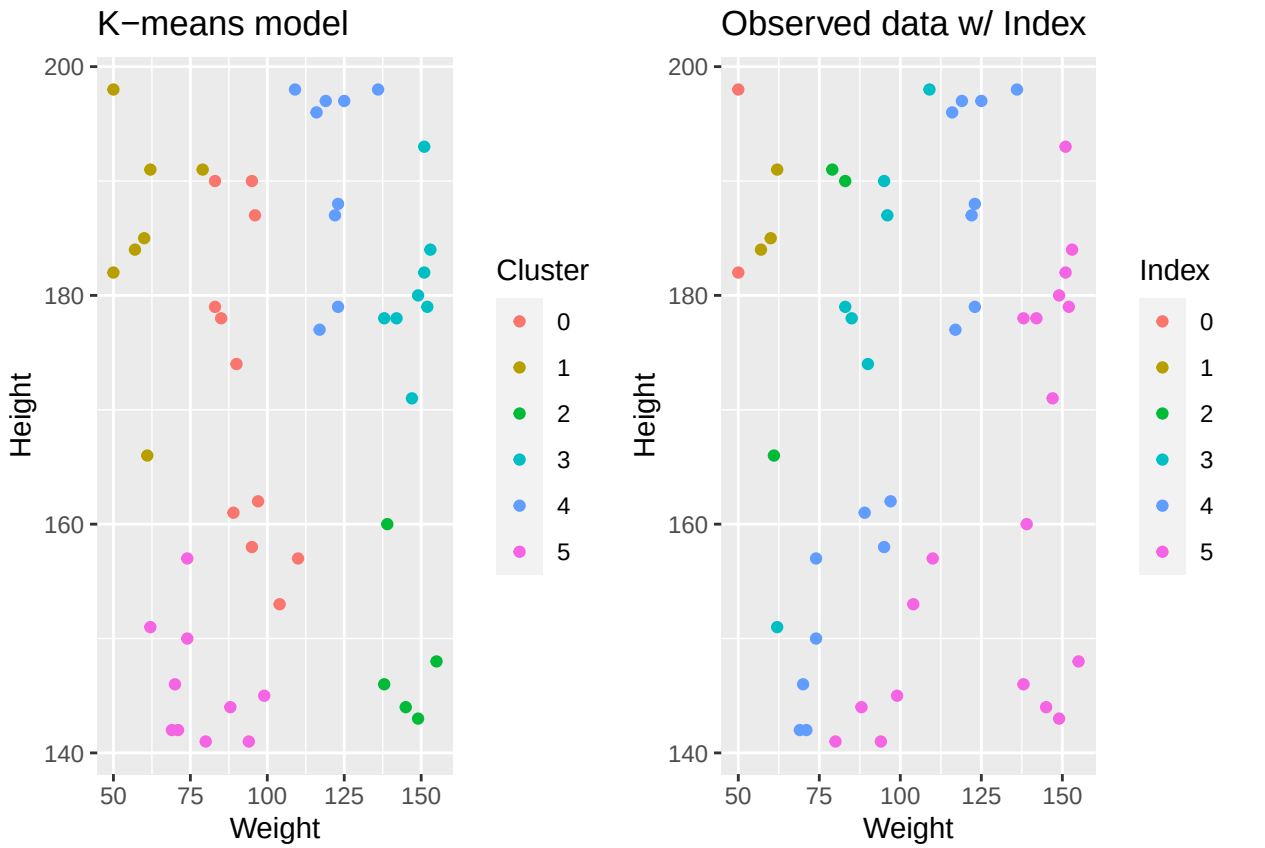
Generally, the ROC curve plots the tradeoff for the true positive rate against the false positive rate as the discrimination threshold varies where the discrimination threshold is the rule to map the predicted probability of being a positive to the label of being a positive. A perfect model gives the above and the worst model would be a diagonal line connecting the origin and (1,1).

Here, I will flip the definitions and regard "males" to be positives so the previous model has a true positive rate of 1. The ROC curve will then be perfect where the true positive rate is always 1. This result follows because the model predicted male with perfect probability so no matter how the discrimination threshold varies, the true positive rate is always 1. See above for this ROC curve.

Problem 5

```
# fit k-means model with k=6
set.seed(68211909)
kmeans_fit = kmeans(dat2 %>% select(Height, Weight), centers = 6)

# plot
kmeans_plot = ggplot(dat2 %>% mutate(kclust = kmeans_fit$cluster-1),
                     aes(x = Weight, y = Height)) +
  geom_point(aes(color = factor(kclust))) +
  scale_color_discrete('Cluster') +
  labs(title = 'K-means model')

dat_plot = ggplot(dat2 %>% mutate(kclust = kmeans_fit$cluster),
                     aes(x = Weight, y = Height)) +
  geom_point(aes(color = factor(Index))) +
  scale_color_discrete('Index') +
  labs(title = 'Observed data w/ Index')

# arrange plots
```

```
grid.arrange(grobs = list(kmeans_plot, dat_plot), ncol = 2)
```



The problem is vague in what it is asking for in terms of "known labels" so I interpret the prompt to mean the "Index" feature in the dataset since that has not been used yet.

In the left-hand side of the above plot, I plotted the observed data (the random 50 males) with the points colored by the clusters given from a k-means model trained on the height and weight features, with k set to 6. The right-hand side shows the observed data with the points colored by the index given in the dataset. The labels for this index are, in order from 0-5, "extremely weak", "weak", "normal", "overweight", "obesity", and "extreme obesity". Looking at these plots side-by-side it does not appear that the clusters given from the k-means model can be interpreted as the index.