Task 4

Task 4: Use 5-fold cross-validation to select the best λ. Compare the prediction performance between the "optimal" model and "full" model.

5-fold CV

We write an **R** function `cv.logit.lasso` to conduct 5-fold cross-validation to select the best λ.

```
cv.logit.lasso <- function(x, y, nfolds = 5, lambda) {
  auc <- data.frame(matrix(ncol = 3, nrow = 0))
  folds <- createFolds(y, k = nfolds)
  for (i in 1:nfolds) {
    valid_index <- folds[[i]]
    x_training <- x[-valid_index, ]
    y_training <- y[-valid_index]
    training_dat <- data.frame(cbind(y_training, x_training))
    x_valid <- cbind(rep(1, length(valid_index)), x[valid_index, ])
    y_valid <- y[valid_index]
    res <- LogisticLASSO(dat = training_dat, start = rep(0, ncol(training_dat)), lambda = lambda)
    for (k in 1:nrow(res)) {
      betavec <- res[k, 2:ncol(res)]
      u_valid <- x_valid %*% betavec
      phat_valid <- sigmoid(u_valid)[, 1]
      roc <- roc(response = y_valid, predictor = phat_valid)
      auc <- rbind(auc, c(lambda[k], i, roc$auc[1]))
    }
  }
  colnames(auc) <- c("lambda", "fold", "auc")
  cv_res <- auc %>%
    group_by(lambda) %>%
    summarize(auc_mean = mean(auc)) %>%
    mutate(auc_ranking = min_rank(desc(auc_mean)))
  bestlambda <- min(cv_res$lambda[cv_res$auc_ranking == 1])
  return(cv_res)
}
```

Compare the results of cross-validation using `glmnet` and using our algorithm.

1. Our function `cv.logit.lasso`:

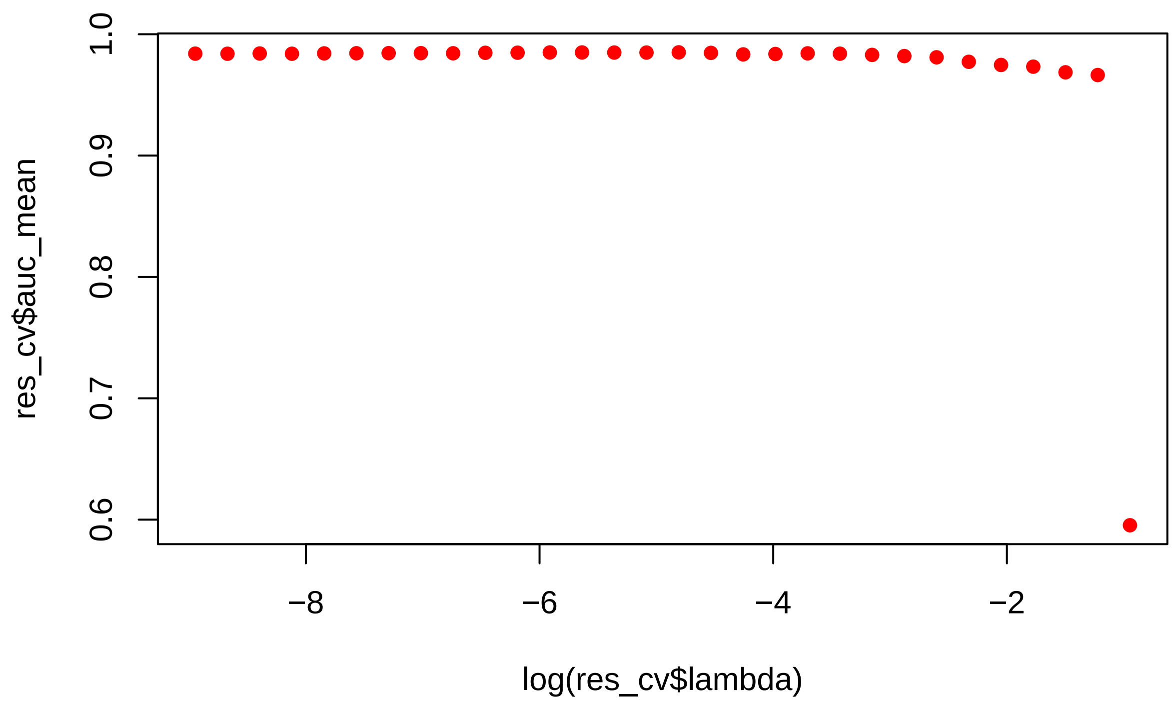
```
lambda_max <- max(t(x) %*% y) / length(y)
lambdas <- exp(seq(log(lambda_max), log(lambda_max) - 8, length = 30))
set.seed(1)
res_cv = cv.logit.lasso(x, y, nfolds = 5, lambda = lambdas)
as.matrix(res_cv %>% arrange(-lambda))
```

```
##              lambda  auc_mean auc_ranking
##  [1,] 0.3880307793 0.5953964          30
##  [2,] 0.2944833885 0.9663792          29
##  [3,] 0.2234886270 0.9686033          28
##  [4,] 0.1696094528 0.9732774          27
##  [5,] 0.1287195992 0.9746754          26
##  [6,] 0.0976875695 0.9772606          25
##  [7,] 0.0741368160 0.9809819          24
##  [8,] 0.0562637346 0.9819983          23
##  [9,] 0.0426995385 0.9829513          22
## [10,] 0.0324054314 0.9839803          17
## [11,] 0.0245930523 0.9842396          13
## [12,] 0.0186641003 0.9837128          20
## [13,] 0.0141645142 0.9833856          21
## [14,] 0.0107496992 0.9846370           8
## [15,] 0.0081581359 0.9850634           1
## [16,] 0.0061913529 0.9848939           4
## [17,] 0.0046987267 0.9848928           5
## [18,] 0.0035659464 0.9849928           3
## [19,] 0.0027062595 0.9849937           2
## [20,] 0.0020538280 0.9847940           6
## [21,] 0.0015586862 0.9846939           7
## [22,] 0.0011829144 0.9842929          12
## [23,] 0.0008977346 0.9844208           9
## [24,] 0.0006813066 0.9844208           9
## [25,] 0.0005170555 0.9843215          11
## [26,] 0.0003924025 0.9842212          14
## [27,] 0.0002978012 0.9839216          18
## [28,] 0.0002260066 0.9841236          15
## [29,] 0.0001715204 0.9839214          19
## [30,] 0.0001301698 0.9840223          16
```

```r
# best lambda
best_lambda <- res_cv$lambda[res_cv$auc_ranking == 1]
best_lambda
```

```
## [1] 0.008158136
```

```r
plot(log(res_cv$lambda), res_cv$auc_mean, pch = 16, col = "red")
```




```r
# coefficients of the best model
res_coef <- LogisticLASSO(dat = Training, start = rep(0, ncol(Training)),
                          lambda = lambdas) %>% as.data.frame
res_coef[res_coef$lambda == best_lambda, -1]
```

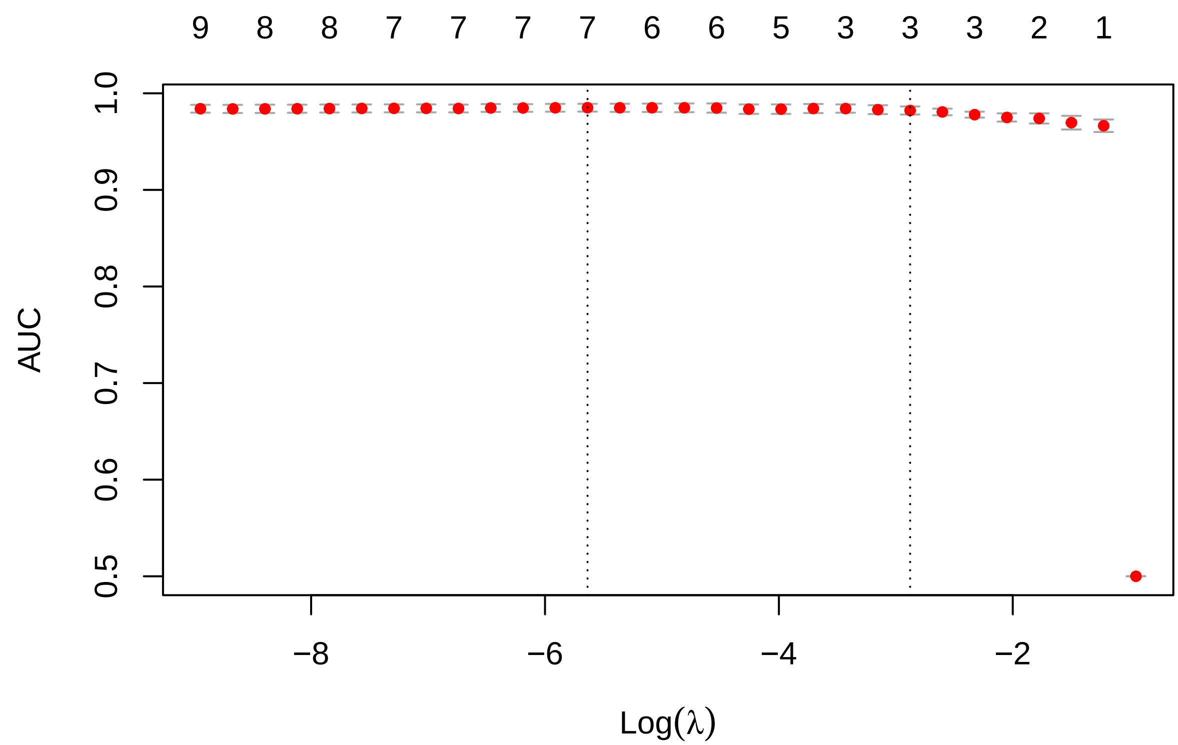
```
##    (Intercept) radius_mean texture_mean perimeter_mean area_mean
## 15   -0.633966   0.3587031     1.005476              0  1.600012
##    smoothness_mean compactness_mean concavity_mean concave.points_mean
## 15       0.2778989                0              0            2.423411
##    symmetry_mean fractal_dimension_mean
## 15     0.3388377                      0
```

2. `glmnet` from **R** package `caret`

```r
set.seed(1)
fit.logit.lasso <- cv.glmnet(x, y,
                             nfolds = 5, alpha = 1,
                             lambda = lambdas,
                             family = "binomial", type.measure = "auc")
# best lambda
fit.logit.lasso$lambda.min
```

```
## [1] 0.003565946
```

```r
plot(fit.logit.lasso)
```




```r
# coefficients of the best model
coef(fit.logit.lasso, fit.logit.lasso$lambda.min)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                  s1
## (Intercept)            -0.59868664
## radius_mean              .
## texture_mean             1.25842810
## perimeter_mean           .
## area_mean                2.80483352
## smoothness_mean          0.56978399
## compactness_mean         .
## concavity_mean           0.19041249
## concave.points_mean      2.23311614
## symmetry_mean            0.46750033
## fractal_dimension_mean -0.03370096
```

The results are slightly different (mean AUC values).

```r
tibble(
  lambda = lambdas,
  ours_AUC = res_cv %>% arrange(-lambda) %>% .$auc_mean,
```

```r
  cv.glmnet_AUC = fit.logit.lasso$cvm
) %>%
  knitr::kable()
```

lambda	ours_AUC	cv.glmnet_AUC
0.3880308	0.5953964	0.5000000
0.2944834	0.9663792	0.9664002
0.2234886	0.9686033	0.9696155
0.1696095	0.9732774	0.9739870
0.1287196	0.9746754	0.9749820
0.0976876	0.9772606	0.9778663
0.0741368	0.9809819	0.9806886
0.0562637	0.9819983	0.9822062
0.0426995	0.9829513	0.9830639
0.0324054	0.9839803	0.9841889
0.0245931	0.9842396	0.9842478
0.0186641	0.9837128	0.9836220
0.0141645	0.9833856	0.9835959
0.0107497	0.9846370	0.9847465
0.0081581	0.9850634	0.9849718
0.0061914	0.9848939	0.9850007
0.0046987	0.9848928	0.9849974
0.0035659	0.9849928	0.9850940
0.0027063	0.9849937	0.9849969
0.0020538	0.9847940	0.9847955
0.0015587	0.9846939	0.9847967
0.0011829	0.9842929	0.9842911
0.0008977	0.9844208	0.9844187
0.0006813	0.9844208	0.9844187
0.0005171	0.9843215	0.9843196
0.0003924	0.9842212	0.9842184
0.0002978	0.9839216	0.9840182
0.0002260	0.9841236	0.9839183
0.0001715	0.9839214	0.9838160
0.0001302	0.9840223	0.9840178

The coefficients of the two final models are different because the best λ's of the two methods are different (but close).

```r
# our best lambda
best_lambda
```

```
## [1] 0.008158136
```

```r
# cv.glmnet's best lambda
fit.logit.lasso$lambda.min
```

```
## [1] 0.003565946
```

```r
tibble(
  predictor = c("(Intercept)", names(Training)[-1]),
  ours_coef = res_coef[res_coef$lambda == best_lambda, -1] %>% as.vector %>% as.numeric,
  cv.glmnet_coef = coef(fit.logit.lasso, fit.logit.lasso$lambda.min) %>% as.vector
) %>%
  knitr::kable()
```

predictor	ours_coef	cv.glmnet_coef
(Intercept)	-0.6339660	-0.5986866
radius_mean	0.3587031	0.0000000
texture_mean	1.0054765	1.2584281
perimeter_mean	0.0000000	0.0000000
area_mean	1.6000115	2.8048335
smoothness_mean	0.2778989	0.5697840
compactness_mean	0.0000000	0.0000000
concavity_mean	0.0000000	0.1904125
concave.points_mean	2.4234109	2.2331161
symmetry_mean	0.3388377	0.4675003
fractal_dimension_mean	0.0000000	-0.0337010

If the best λ's are the same (here we take the best λ of `cv.glmnet`), the coefficients are very similar but still slightly different. This difference may cause the slight difference of the CV results (mean AUC), and thus the λ's that has the largest mean AUC values of the two methods are not the same. (Or maybe due to different 5 folds or other reasons)

```r
# use cv.glmnet's best lambda
tibble(
  predictor = c("(Intercept)", names(Training)[-1]),
  ours_coef = res_coef[res_coef$lambda == fit.logit.lasso$lambda.min, -1] %>% as.vector %>% as.numeric,
  cv.glmnet_coef = coef(fit.logit.lasso, fit.logit.lasso$lambda.min) %>% as.vector
) %>%
  knitr::kable()
```

predictor	ours_coef	cv.glmnet_coef
(Intercept)	-0.5995736	-0.5986866
radius_mean	0.0000000	0.0000000
texture_mean	1.2552028	1.2584281
perimeter_mean	0.0000000	0.0000000
area_mean	2.7887923	2.8048335
smoothness_mean	0.5743011	0.5697840
compactness_mean	0.0000000	0.0000000
concavity_mean	0.2001204	0.1904125
concave.points_mean	2.2228355	2.2331161
symmetry_mean	0.4654458	0.4675003
fractal_dimension_mean	-0.0423927	-0.0337010

Prediction performance comparison

We probably need resampling methods (conducted in training data) to select the best model. Is the resampling methods in task 2 correct?

Below is the prediction performance on the test data. *(I suppose this should not be used for model comparison)*

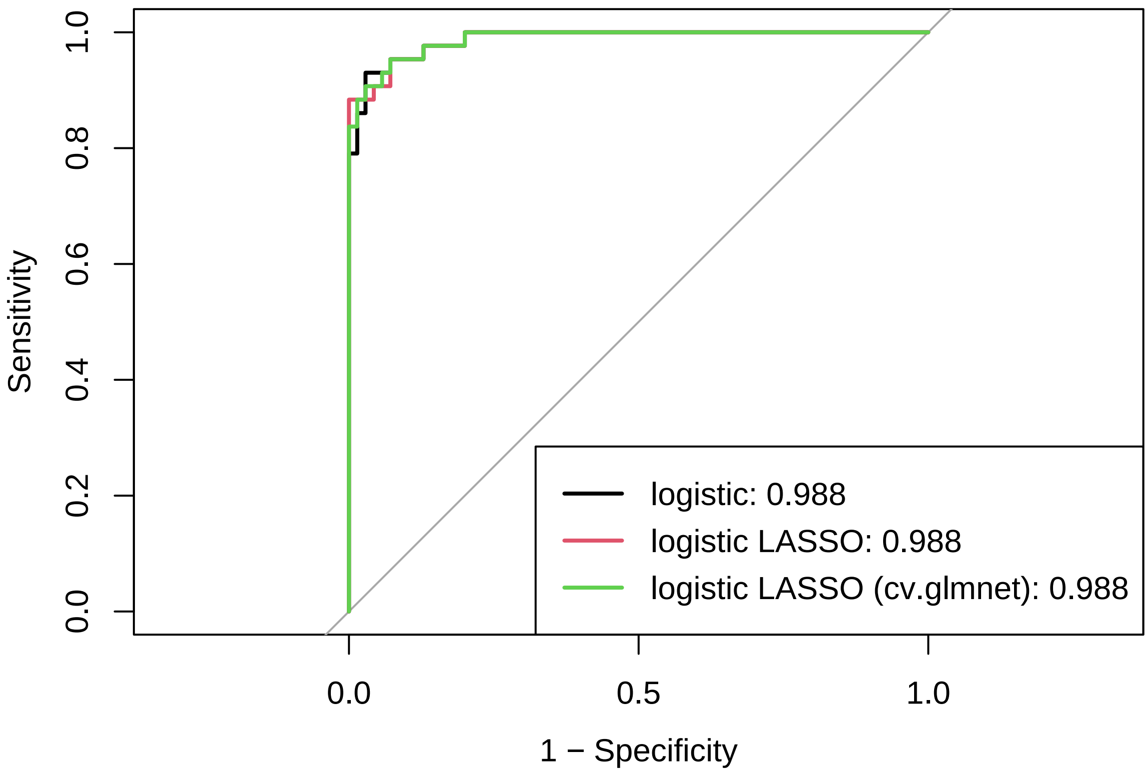
```r
# test data
X_test <- cbind(rep(1, nrow(Test)), model.matrix(diagnosis ~ ., Test)[, -1])
y_test <- Test$diagnosis

# logistic model
res_logit <- NewtonRaphson(dat = Training, func = logisticstuff, start = rep(0, ncol(Training)))
betavec_logit <- res_logit[nrow(res_logit), 3:ncol(res_logit)]
u <- X_test %*% betavec_logit
phat <- sigmoid(u)[, 1]
roc.logit <- roc(response = y_test, predictor = phat)

# logistic LASSO model
betavec_logit.lasso <- res_coef[res_coef$lambda == best_lambda, -1] %>% as.vector %>% as.numeric
u <- X_test %*% betavec_logit.lasso
phat <- sigmoid(u)[, 1]
roc.logitlasso <- roc(response = y_test, predictor = phat)

# logistic LASSO model (cv.glmnet)
betavec_logit.lasso.glm <- coef(fit.logit.lasso, fit.logit.lasso$lambda.min) %>% as.vector
u <- X_test %*% betavec_logit.lasso.glm
phat <- sigmoid(u)[, 1]
roc.logitlasso.glm <- roc(response = y_test, predictor = phat)

# draw rocs
auc <- c(roc.logit$auc[1], roc.logitlasso$auc[1], roc.logitlasso.glm$auc[1])
plot(roc.logit, legacy.axes = TRUE)
plot(roc.logitlasso, col = 2, add = TRUE)
plot(roc.logitlasso.glm, col = 3, add = TRUE)
modelNames <- c("logistic", "logistic LASSO", "logistic LASSO (cv.glmnet)")
legend("bottomright", legend = paste0(modelNames, ": ", round(auc, 3)),
col = 1:3, lwd = 2)
```


Sensitivity
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1 − Specificity
logistic: 0.988
logistic LASSO: 0.988
logistic LASSO (cv.glmnet): 0.988