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`:

```
set.seed(1)
folds <- createFolds(y, k = 5)
lambda_max_i <- rep(NA, 5) # lambda_max for each training set in CV (4/5 of the whole training data)
for (i in 1:5) {
    valid_index <- folds[[i]]
```

```
    x_training <- x[-valid_index, ]
    y_training <- y[-valid_index]
    lambda_max_i[i] <- max(abs(t(x_training) %*% (y_training - mean(y_training)))) / length(y_training)
    }
lambda_max <- max(lambda_max_i) + 1e-10 # max of lambda_max's so that all beta_i's = 0 except intercept
lambdas <- exp(seq(log(lambda_max), log(lambda_max) - 6, length = 30))

set.seed(1)
res_cv = cv.logit.lasso(x, y, nfolds = 5, lambda = lambdas)
as.matrix(res_cv %>% arrange(-lambda))
```

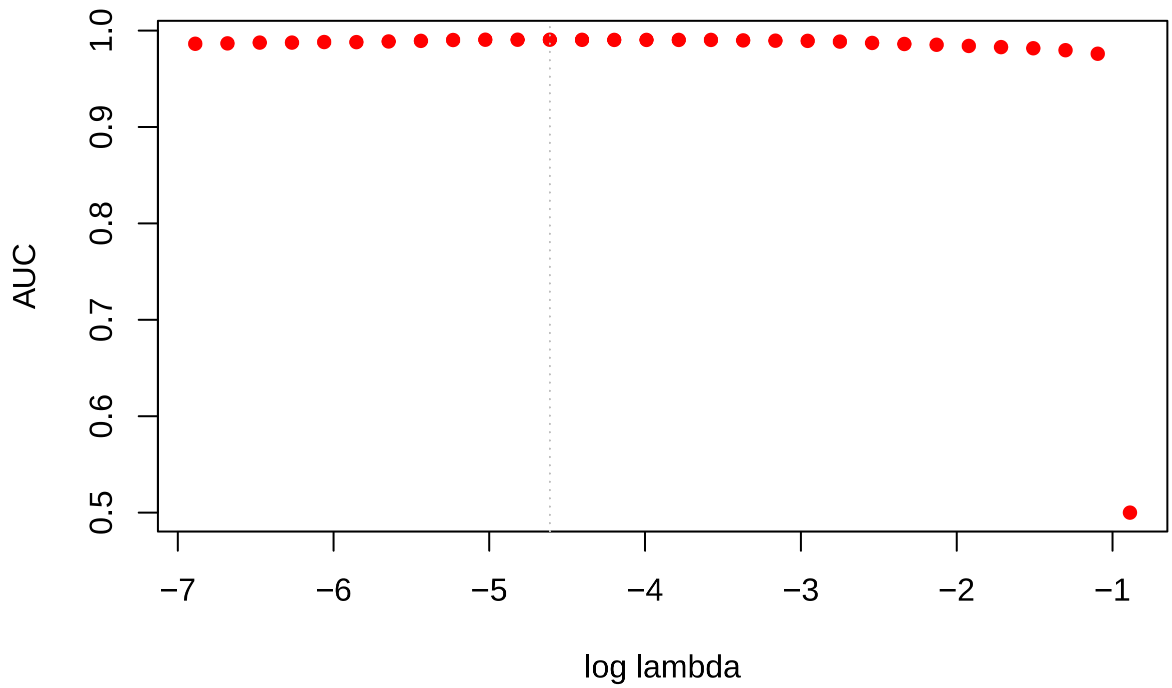
```
##            lambda  auc_mean auc_ranking
##  [1,] 0.411544526 0.5000000          30
##  [2,] 0.334628402 0.9760054          29
##  [3,] 0.272087611 0.9796678          28
##  [4,] 0.221235460 0.9816903          27
##  [5,] 0.179887384 0.9828852          26
##  [6,] 0.146267108 0.9841071          25
##  [7,] 0.118930336 0.9853029          24
##  [8,] 0.096702703 0.9861031          23
##  [9,] 0.078629332 0.9872060          20
## [10,] 0.063933805 0.9885875          15
## [11,] 0.051984817 0.9893408          12
## [12,] 0.042269050 0.9895397          11
## [13,] 0.034369124 0.9898394          10
## [14,] 0.027945664 0.9903360           5
## [15,] 0.022722724 0.9903354           6
## [16,] 0.018475933 0.9903346           8
## [17,] 0.015022850 0.9903347           7
## [18,] 0.012215136 0.9904332           4
## [19,] 0.009932173 0.9905325           1
## [20,] 0.008075887 0.9905317           3
## [21,] 0.006566534 0.9905324           2
## [22,] 0.005339274 0.9902345           9
## [23,] 0.004341384 0.9893390          13
## [24,] 0.003529995 0.9887423          14
## [25,] 0.002870252 0.9880447          17
## [26,] 0.002333813 0.9881480          16
## [27,] 0.001897632 0.9875518          18
## [28,] 0.001542972 0.9875498          19
## [29,] 0.001254596 0.9867502          21
## [30,] 0.001020117 0.9863500          22
```

```
# best lambda
best_lambda <- max(res_cv$lambda[res_cv$auc_ranking == 1])
best_lambda
```

```
## [1] 0.009932173
```

```
plot(log(res_cv$lambda),
     res_cv$auc_mean,
     pch = 16,
```

```r
     xlab = "log lambda",
     ylab = "AUC",
     col = "red")
abline(v = log(res_cv$lambda[which((res_cv$auc_ranking == 1))]), col = "gray", lty = 3)
```




```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]
```

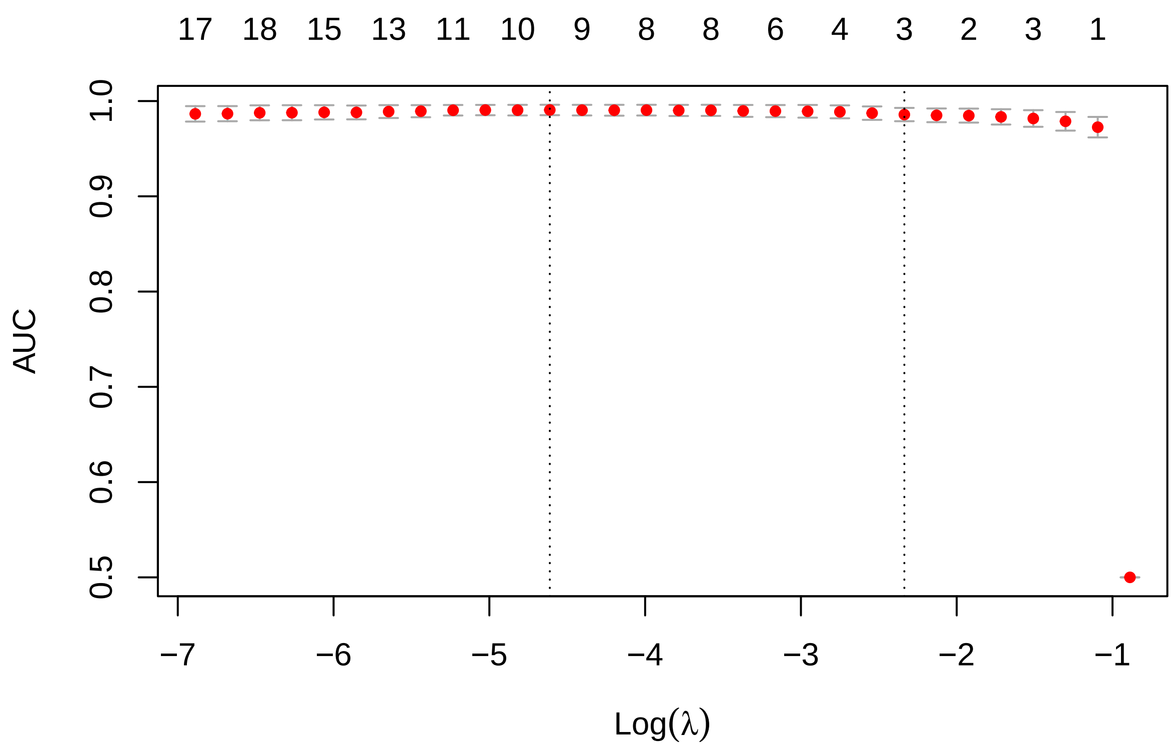
```
##    (Intercept) radius_mean texture_mean perimeter_mean area_mean
## 19   -0.613828           0    0.3958698              0         0
##    smoothness_mean compactness_mean concavity_mean concave.points_mean
## 19               0                0              0           0.4850029
##    symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 19             0                      0 0.7660262          0            0
##    area_se smoothness_se compactness_se concavity_se concave.points_se
## 19       0             0              0            0                 0
##    symmetry_se fractal_dimension_se radius_worst texture_worst perimeter_worst
## 19           0          -0.03644811     2.587817     0.4394404               0
##    area_worst smoothness_worst compactness_worst concavity_worst
## 19          0        0.4980005                 0        0.143706
##    concave.points_worst symmetry_worst fractal_dimension_worst
## 19             1.008792      0.3617036                       0
```

2. glmnet from **R** package caret

```
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] 0.009932173
```

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




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

```
## 31 x 1 sparse Matrix of class "dgCMatrix"
##                               s1
## (Intercept)          -0.60486590
## radius_mean            .
## texture_mean          0.38113717
## perimeter_mean         .
## area_mean              .
```

```
## smoothness_mean          .
## compactness_mean         .
## concavity_mean           .
## concave.points_mean      0.44764990
## symmetry_mean            .
## fractal_dimension_mean   .
## radius_se                0.83791365
## texture_se               .
## perimeter_se             .
## area_se                  .
## smoothness_se            .
## compactness_se           .
## concavity_se             .
## concave.points_se        .
## symmetry_se              .
## fractal_dimension_se    -0.03686225
## radius_worst             2.57346076
## texture_worst            0.45902449
## perimeter_worst          .
## area_worst               .
## smoothness_worst         0.49454577
## compactness_worst        .
## concavity_worst          0.12532306
## concave.points_worst     1.06926848
## symmetry_worst           0.36416342
## fractal_dimension_worst  .
```

The results are slightly different (mean AUC values).

```
tibble(
  lambda = lambdas,
  ours_AUC = res_cv %>% arrange(-lambda) %>% .$auc_mean,
  cv.glmnet_AUC = fit.logit.lasso$cvm
) %>%
  knitr::kable()
```

lambda	ours_AUC	cv.glmnet_AUC
0.4115445	0.5000000	0.5000000
0.3346284	0.9760054	0.9725915
0.2720876	0.9796678	0.9787526
0.2212355	0.9816903	0.9817053
0.1798874	0.9828852	0.9834002
0.1462671	0.9841071	0.9847216
0.1189303	0.9853029	0.9850192
0.0967027	0.9861031	0.9858166
0.0786293	0.9872060	0.9873215
0.0639338	0.9885875	0.9887026
0.0519848	0.9893408	0.9892575
0.0422691	0.9895397	0.9894566
0.0343691	0.9898394	0.9896556
0.0279457	0.9903360	0.9902522
0.0227227	0.9903354	0.9901527

lambda	ours_AUC	cv.glmnet_AUC
0.0184759	0.9903346	0.9904508
0.0150229	0.9903347	0.9903518
0.0122151	0.9904332	0.9904501
0.0099322	0.9905325	0.9906491
0.0080759	0.9905317	0.9904501
0.0065665	0.9905324	0.9905479
0.0053393	0.9902345	0.9903485
0.0043414	0.9893390	0.9893569
0.0035300	0.9887423	0.9889618
0.0028703	0.9880447	0.9880654
0.0023338	0.9881480	0.9881674
0.0018976	0.9875518	0.9877693
0.0015430	0.9875498	0.9876669
0.0012546	0.9867502	0.9867649
0.0010201	0.9863500	0.9865655

The best λ's are the same, and the coefficients are very similar.

```
# our best lambda
best_lambda
```

```
## [1] 0.009932173
```

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

```
## [1] 0.009932173
```

```
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)	-0.6138280	-0.6048659
radius_mean	0.0000000	0.0000000
texture_mean	0.3958698	0.3811372
perimeter_mean	0.0000000	0.0000000
area_mean	0.0000000	0.0000000
smoothness_mean	0.0000000	0.0000000
compactness_mean	0.0000000	0.0000000
concavity_mean	0.0000000	0.0000000
concave.points_mean	0.4850029	0.4476499
symmetry_mean	0.0000000	0.0000000
fractal_dimension_mean	0.0000000	0.0000000
radius_se	0.7660262	0.8379136

predictor	ours_coef	cv.glmnet_coef
texture_se	0.0000000	0.0000000
perimeter_se	0.0000000	0.0000000
area_se	0.0000000	0.0000000
smoothness_se	0.0000000	0.0000000
compactness_se	0.0000000	0.0000000
concavity_se	0.0000000	0.0000000
concave.points_se	0.0000000	0.0000000
symmetry_se	0.0000000	0.0000000
fractal_dimension_se	-0.0364481	-0.0368622
radius_worst	2.5878169	2.5734608
texture_worst	0.4394404	0.4590245
perimeter_worst	0.0000000	0.0000000
area_worst	0.0000000	0.0000000
smoothness_worst	0.4980005	0.4945458
compactness_worst	0.0000000	0.0000000
concavity_worst	0.1437060	0.1253231
concave.points_worst	1.0087922	1.0692685
symmetry_worst	0.3617036	0.3641634
fractal_dimension_worst	0.0000000	0.0000000

Prediction performance comparison

Below is the prediction performance on the test data.

```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)))
```

```
## Warning in NewtonRaphson(dat = Training, func = logisticstuff, start = rep(0, :
## Complete separation occurs. Algorithm does not converge.
```

```r
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, -c(1, 2)]
col_nonzero <- names(betavec_logit.lasso)[betavec_logit.lasso != 0]
df_nonzero <- Training[c("diagnosis", col_nonzero)]
refit_logit <- NewtonRaphson(dat = df_nonzero, func = logisticstuff, start = rep(0, ncol(df_nonzero)))
betavec_lasso.refit <- refit_logit[nrow(refit_logit), 3:ncol(refit_logit)]
betavec_lasso.refit <- bind_rows(betavec_logit.lasso, betavec_lasso.refit)[2,] %>% select("(Intercept)"
betavec_lasso.refit[is.na(betavec_lasso.refit)] <- 0
betavec_lasso.refit <- as.numeric(betavec_lasso.refit)
u <- X_test %*% betavec_lasso.refit
phat <- sigmoid(u)[, 1]
```

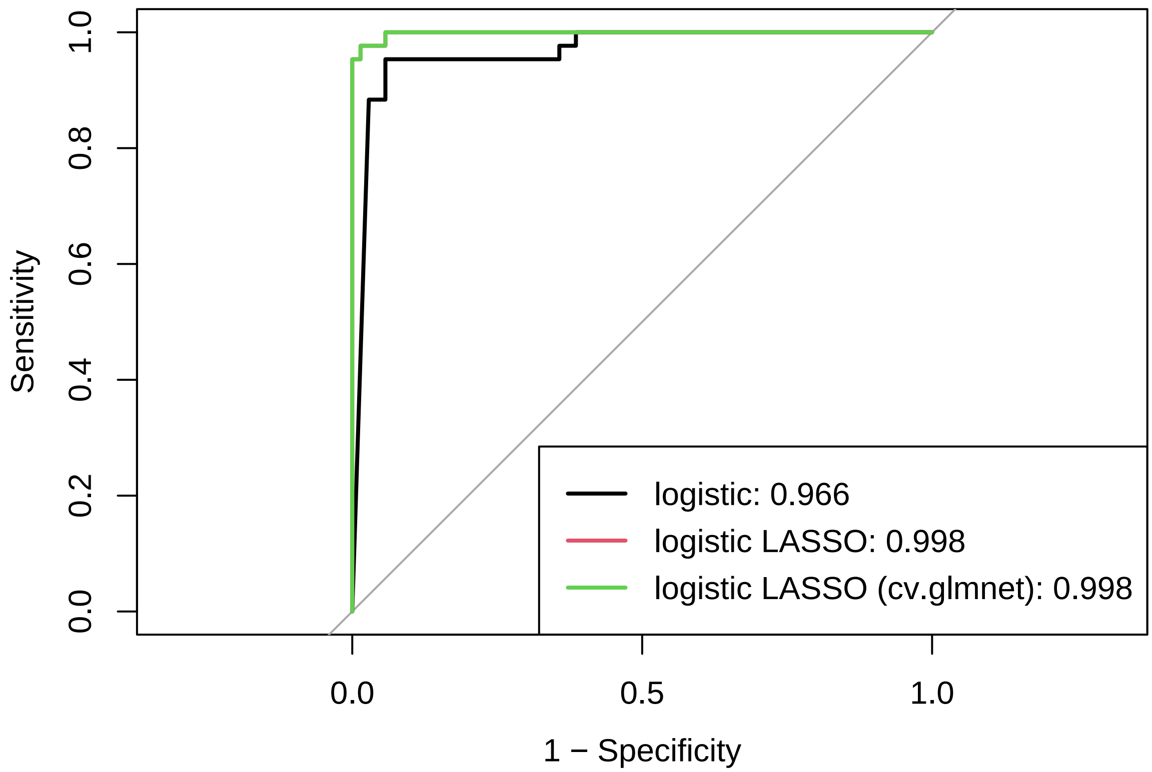
```
roc.logitlasso <- roc(response = y_test, predictor = phat)

# logistic LASSO model (cv.glmnet)
betavec_logit.lasso.glm_temp <- coef(fit.logit.lasso, fit.logit.lasso$lambda.min)
betavec_logit.lasso.glm <- betavec_logit.lasso.glm_temp %>% as.vector
names(betavec_logit.lasso.glm) <- betavec_logit.lasso.glm_temp@Dimnames[[1]]
col_nonzero <- names(betavec_logit.lasso.glm)[betavec_logit.lasso.glm != 0][-1]
df_nonzero <- Training[c("diagnosis", col_nonzero)]
refit_logit.glm <- NewtonRaphson(dat = df_nonzero, func = logisticstuff, start = rep(0, ncol(df_nonzero)
betavec_lasso.refit.glm <- refit_logit.glm[nrow(refit_logit.glm), 3:ncol(refit_logit.glm)]
betavec_lasso.refit.glm <- bind_rows(betavec_logit.lasso, betavec_lasso.refit.glm)[2,] %>% select("(Int
betavec_lasso.refit.glm[is.na(betavec_lasso.refit.glm)] <- 0
betavec_lasso.refit.glm <- as.numeric(betavec_lasso.refit.glm)
u <- X_test %*% betavec_lasso.refit.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])
auc
```

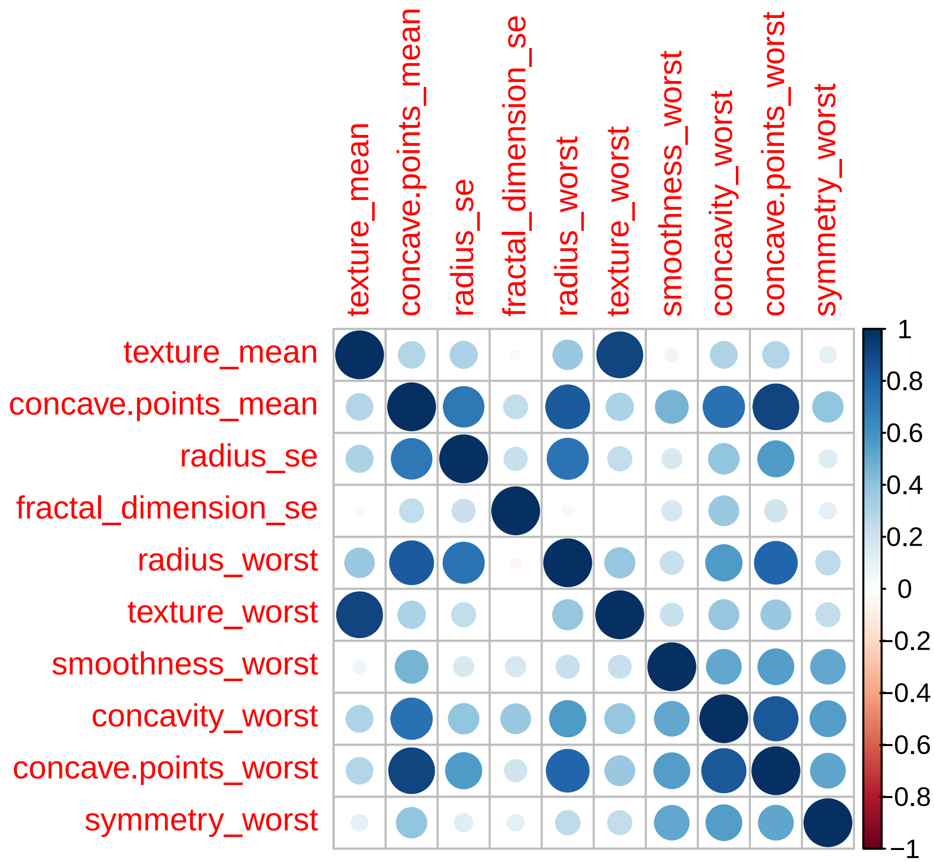
```
## [1] 0.9661130 0.9983389 0.9983389
```

```
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)
```



Correlation Plot

```
corrplot::corrplot(cor(Training[names(betavec_logit.lasso)[betavec_logit.lasso != 0]]))
```



LASSO model coefficients

Re-fit the logistic regression with the predictors selected by LASSO.

```r
refit_logit[nrow(refit_logit), 3:ncol(refit_logit)]
```

```
##          (Intercept)         texture_mean  concave.points_mean
##          -0.09092737           0.94392378           0.84754704
##            radius_se fractal_dimension_se         radius_worst
##           3.77040802          -1.12476354           6.01510669
##        texture_worst     smoothness_worst      concavity_worst
##           1.23805580           1.67539455           1.19425400
## concave.points_worst       symmetry_worst
##           1.87249528           0.69899461
```