DUPLICATE: Full NR and LASSO Run (with stadardized data)

Jimmy Kelliher

3/27/2022

This file prepares our data, runs Newton-Raphson, runs LASSO, and plots some results at the bottom. I think this file should stay clean and relatively simple. I think the file should only take 15ish minutes to knit or run in full. I think the best idea is for plots to be saved as R objects and imported into the presentation / report, but open to ideas here.

```
source("./shared_code/setup.R")
knitr::opts_chunk$set(cache = TRUE)
```

```
source("./shared_code/data_prep.R")
```

Newton-Raphson

```
source("./shared_code/full_NR2.R")
source("./shared_code/roc_func.R")
roc_nr <- roc_func(nr_beta_est, bc_tst)
```

LASSO

```
source("./shared_code/logistic_lasso.R")
source("./shared_code/auc_calc_lasso.R")
```

Folding Data

```
source("./shared_code/cv_folding.R")

bc_trn_folds <-
  cv_sets(training = bc_trn) %>%
  select(-fold_p)
```

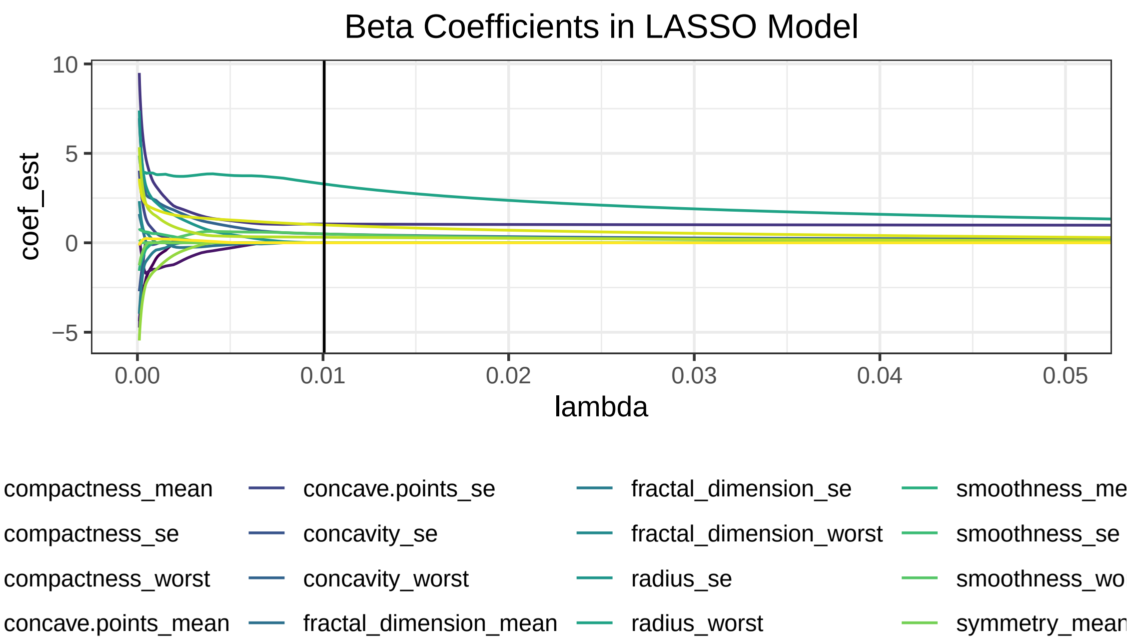
```
source("./shared_code/cv_implementation.R")
lambda_max <- max(t(scale(as.matrix(bc_trn[,-1]))) %*% bc_trn[,1]) / nrow(bc_trn[,-1])
lambda_list <- list(log(lambda_max), log(0.0001), -(log(lambda_max) - log(0.0001))/100, exp)

#cv_res <- cv_jt(k = 5, training = bc_trn_folds,
#                func = logistic_lasso, lam_start_stop_func = log,
#                lambda_list = lambda_list)
```

```r
# cv.glmnet(x, y, alpha = 1, family = "binomial", type.measure = "auc", lambda = exp(seq(log(lambda_max
```

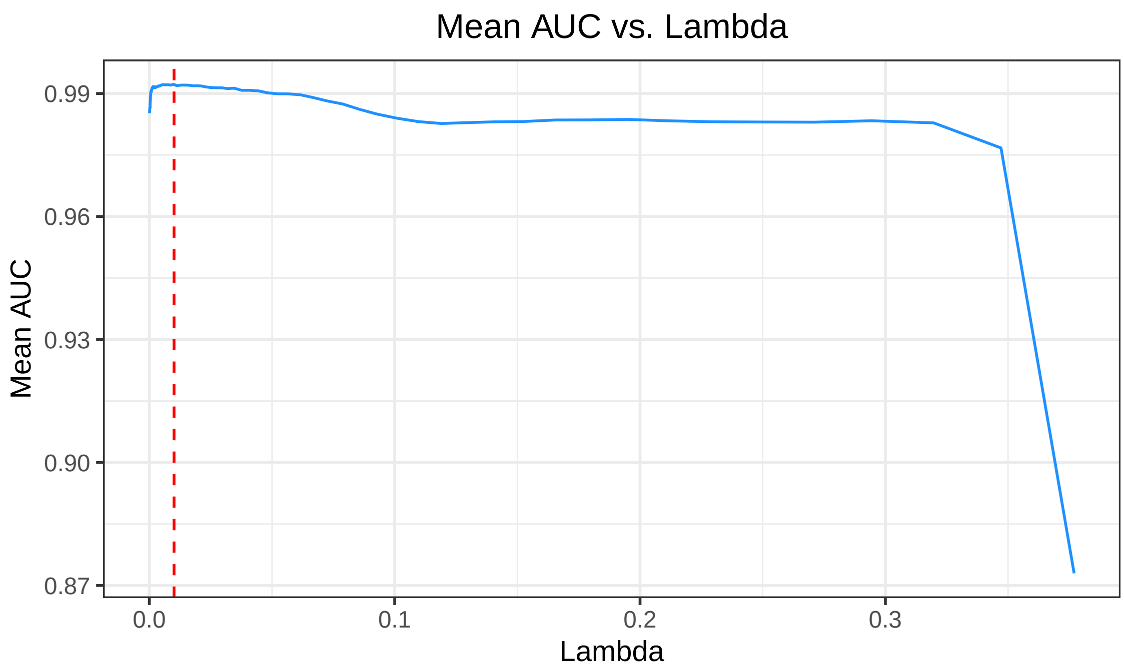
```r
# load data from final run
load("finalized_lasso_data.RData")

# pulling out key results
selected_lambda <-
  cv_res[[1]][[length(cv_res[[1]])]] %>%
  filter(mean_auc == max(mean_auc)) %>%
  pull(lambda) %>%
  min()

selected_lambda_minmax <-
  cv_res[[3]] %>%
  group_by(lambda) %>%
  summarize(min_auc = min(auc_vals)) %>%
  ungroup() %>%
  filter(min_auc == max(min_auc)) %>%
  pull(lambda) %>%
  as.vector()



lasso_final_model <- logistic_lasso(inputs = bc_trn[,-1],
                                    output = bc_trn[,1],
                                    lambda_vec = selected_lambda)
lasso_betas <- lasso_final_model[[2]] %>% t()
# this is just for viz purposes
tst_lambda_vec <- exp(seq(from = log(lambda_max),
                          to = log(0.0001),
                          by = -(log(lambda_max) - log(0.0001))/100))

#lasso_final_range <- logistic_lasso(inputs = bc_trn[,-1],
#                                     output = bc_trn[,1],
#                                     lambda_vec = tst_lambda_vec)

lfr_df <- data.frame(do.call(cbind, lasso_final_range)) %>%
  select(-selected) %>%
  pivot_longer(cols = starts_with("beta"),
               names_prefix = "beta.",
               names_to = "beta_coef",
               values_to = "coef_est")

# storing lasso ROC object
bc_tst[,-1] <- scale(bc_tst[,-1])
roc_lasso <- roc_func(lasso_betas, bc_tst)

cv_res_lam <- cv_res[[1]]
```

```r
# each of these plots could be saved as R objects and imported into other documents
lfr_df %>%
  group_by(lambda) %>%
  filter(beta_coef != "intercept") %>%
  ggplot(x = lambda, y = coef_est, group = beta_coef) +
```

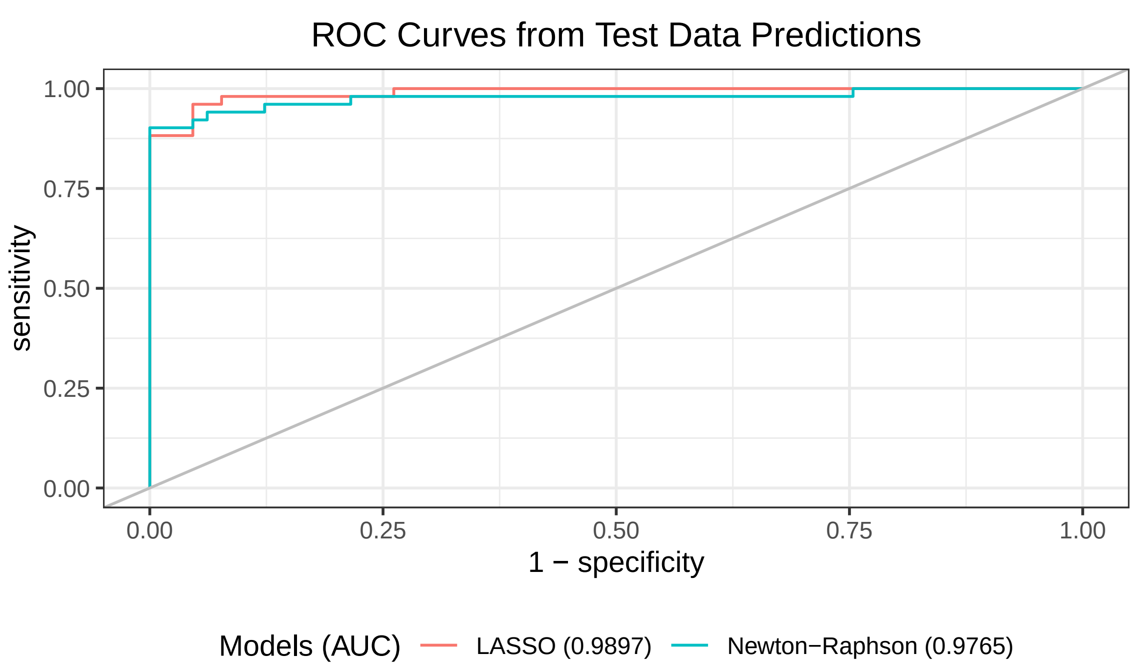
```r
  geom_path(aes(x = lambda, y = coef_est, group = beta_coef, col = beta_coef)) +
  coord_cartesian(xlim = c(0, 0.05)) + # this zooms in on the plot, comment out if flipping
#  scale_x_reverse() +                   # this flips the x-axis if wanted
#  coord_cartesian(xlim = c(0.06, 0)) + # this zooms in if flipped
  geom_vline(xintercept = selected_lambda) +
  labs(title = "Beta Coefficients in LASSO Model")
```




```r
# this looks alright, might prefer a larger lambda range to show AUC down to 0.5
# but probably not important
cv_res[[1]][[1]] %>%
  data.frame() %>%
  ggplot(x = lambda, y = mean_auc) +
  geom_line(aes(x = lambda, y = mean_auc), col = "dodgerblue") +
  geom_vline(xintercept = selected_lambda, linetype = "dashed", col = "red") +
  labs(title = "Mean AUC vs. Lambda",
       x = "Lambda",
       y = "Mean AUC")
```

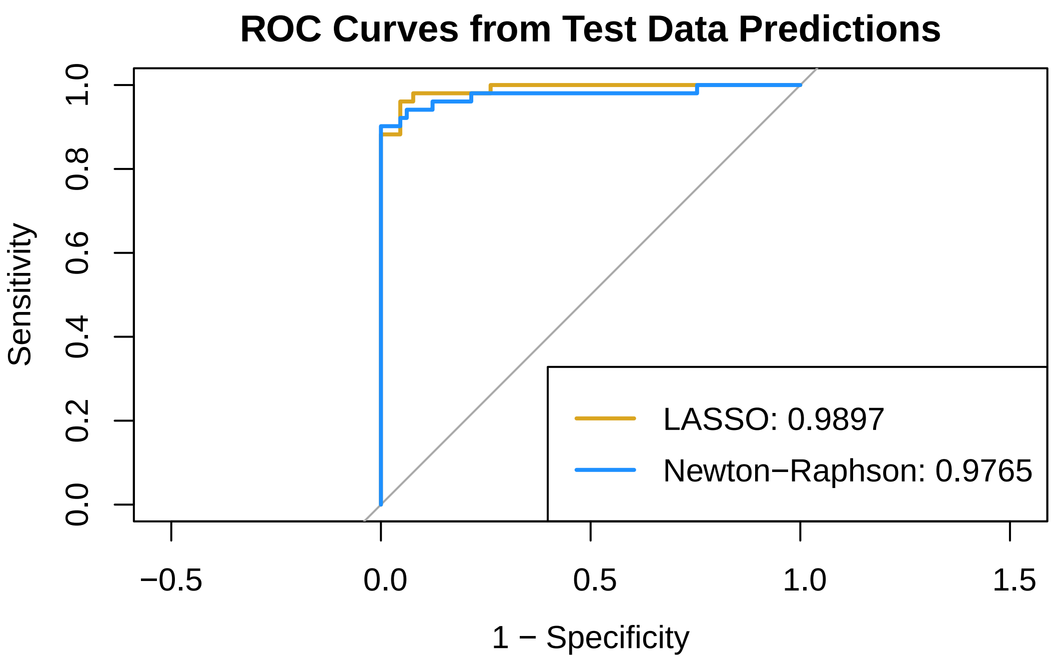



```r
# Final ROC plots
auc_vec <- c(round(roc_lasso$auc[1], digits = 4), round(roc_nr$auc[1], digits = 4))
model_names <- c("LASSO", "Newton-Raphson")
ggroc(list(roc_lasso, roc_nr), legacy.axes = TRUE) +
  scale_color_discrete(labels = paste0(model_names, " (", auc_vec, ")"),
                       name = "Models (AUC)") +
  geom_abline(intercept = 0, slope = 1, color = "grey") +
  labs(title = "ROC Curves from Test Data Predictions")
```




```r
## another option for plotting ROC curves

plot(roc_lasso, legacy.axes = TRUE, col = "goldenrod",
```

```
     main = "ROC Curves from Test Data Predictions")
plot(roc_nr, col = "dodgerblue", add = TRUE)
legend("bottomright", legend = paste0(model_names, ": ", auc_vec),
       col = c("goldenrod", "dodgerblue"), lwd = 2)
```




```
beta_names <-
  (dimnames(lasso_betas)[1]) %>%
  data.frame() %>%
  rename(beta_coef = 1)

cbind(nr_beta_est, lasso_betas) %>%
  data.frame() %>%
  round(digits = 4) %>%
  rename(NewtonRaphson = 1, LASSO = 2) %>%
  knitr::kable(caption = "Final Beta Coefficient Estimates")
```

Table 1: Final Beta Coefficient Estimates

	NewtonRaphson	LASSO
intercept	-0.0881	-0.9302
texture_mean	8.2691	0.9994
smoothness_mean	-2.5447	0.0000
compactness_mean	-10.9297	0.0000
concave points_mean	20.9342	1.0512
symmetry_mean	-1.9716	0.0000
fractal_dimension_mean	3.9221	0.0000
radius_se	18.5308	0.0000
texture_se	-2.0504	0.0000
smoothness_se	1.2753	0.0000
compactness_se	4.7358	0.0000
concavity_se	-6.4914	0.0000
concave points_se	8.9092	0.0000
symmetry_se	-13.8273	0.0000

	NewtonRaphson	LASSO
fractal_dimension_se	-12.2152	0.0000
radius_worst	9.2654	3.2848
smoothness_worst	0.6705	0.4955
compactness_worst	-14.3912	0.0000
concavity_worst	14.9611	0.4943
symmetry_worst	12.3102	0.3112
fractal_dimension_worst	7.7655	0.0000

Comparison to GLMNET Lasso

```r
# standardize bc_trn[,-1]
bc_cov <- as.matrix(scale(bc_trn[,-1]))

glmnet_fit <- glmnet(x = bc_cov , y = bc_trn[,1], family = "binomial", lambda = selected_lambda)
glmnet_est <- as.vector(coef(glmnet_fit))

comp_est <- tibble(
  names = lasso_betas %>% rownames(),
  glmnet_est = glmnet_est,
  lasso_est = lasso_betas,
  diff = glmnet_est - lasso_est
)
comp_est %>% knitr::kable()
```

names	glmnet_est	lasso_est	diff
intercept	-0.9302305	-0.9301544	-7.606137e-05
texture_mean	0.9999167	0.9994411	4.755628e-04
smoothness_mean	0.0000000	0.0000000	0.000000e+00
compactness_mean	0.0000000	0.0000000	0.000000e+00
concave points_mean	1.0522363	1.0512135	1.022828e-03
symmetry_mean	0.0000000	0.0000000	0.000000e+00
fractal_dimension_mean	0.0000000	0.0000000	0.000000e+00
radius_se	0.0000000	0.0000000	0.000000e+00
texture_se	0.0000000	0.0000000	0.000000e+00
smoothness_se	0.0000000	0.0000000	0.000000e+00
compactness_se	0.0000000	0.0000000	0.000000e+00
concavity_se	0.0000000	0.0000000	0.000000e+00
concave points_se	0.0000000	0.0000000	0.000000e+00
symmetry_se	0.0000000	0.0000000	0.000000e+00
fractal_dimension_se	0.0000000	0.0000000	0.000000e+00
radius_worst	3.2856396	3.2847603	8.792463e-04
smoothness_worst	0.4956609	0.4954759	1.849726e-04
compactness_worst	0.0000000	0.0000000	0.000000e+00
concavity_worst	0.4941184	0.4942962	-1.777750e-04
symmetry_worst	0.3113843	0.3112113	1.730019e-04
fractal_dimension_worst	0.0000000	0.0000000	0.000000e+00

The GLMNET lasso regression procedure and our logistic lasso procedure produce very similar (near identical) results. Yay!