Computing Group Project 2

Muhire Kwizera, Quinton Neville, Zelos Zhu

March 2019

1 Introduction

While cancer incidence rates have been marginally increasing over time, death rates have been decreasing in recent years. Speculatively, there are many aspects of the medical process which may be contributing to this trend, but the foremost predictors of improved outcomes are well known to be early detection and diagnosis. As such, statistical learning in the realms of classification and decision have become a natural part of the diagnostic process. With respect to these breast cancer data, specifically, we built and compared a Newton-Raphson algorithm and the Pathwise Coordinate-descent optimization algorithm with LASSO regularization for the estimation of logistic regression covariates for binary malignant or benign tumor classification.

2 Methods

2.1 Data

The data in this study consisted of breast cancer diagnoses for 569 individuals and 32 variables. The outcome of interest, breast cancer diagnosis, was binary (malignant or benign) and the predictors (continuous) were the means, standard errors, and largest values of several features of the subjects' breast tissue images (radius, texture, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension).[1] The additional variable was a unique identification number for each de-identified subject.

2.2 Statistical Analysis

We explored the relationship between the outcome and the predictors using box plots, and we investigated associations among the predictors using the Spearman's rank correlation coefficient. Variable selection was based on the results from these exploratory data analyses. All the predictors were centered and scaled around their corresponding sample means and sample standard deviations respectively.

[1] UCI Machine Learning Repository: Breast Cancer Wisconsin (Diagnostic) Data Set. Web

We fitted the logistic model (1) - (2) to classify the breast tissue images into the malignant or benign categories. Y_i and x_i were the i^{th} subject's breast cancer diagnosis and vector of predictors respectively.

$$Y_i \sim \text{Bernoulli}(p_i) \tag{1}$$

After deriving the required components, the Log-likelihood, Gradient, and Hessian are derived such that, for

$$F(\mathbf{x_i}^T\beta) = P(Y_i = 1|\beta) = \frac{\mathrm{e}^{\mathbf{x}_i^T\beta}}{1+\mathrm{e}^{\mathbf{x}_i^T\beta}} = p_i \tag{2}$$

- **Log-Likelihood**

$$\mathcal{L}(Y_i|\beta) = \sum_{i=1}^{n}\left(Y_i\left(\mathbf{x}_i^T\beta\right) - log\left(1+\mathrm{e}^{\mathbf{x}_i^T\beta}\right)\right) \tag{3}$$

- **Gradient**

  given $p = 30+1 = 31$, for the intercept, and let $p_i = P(Y_i = 1|x_i, \beta)$.

$$\begin{aligned}\nabla f(Y_i|(\beta_0,\ldots,\beta_{p-1}) &= \begin{pmatrix} \sum_{i=1}^n x_{i,0}(y_i - p_i) \\ \sum_{i=1}^n x_{i,1}(y_i - p_i) \\ \vdots \\ \sum_{i=1}^n x_{i,p-1}(y_i - p_i) \end{pmatrix} \\ &= \sum_{i=0}^{p-1} \mathbf{x_i}^T(Y_i - F(\mathbf{x}_i^T\beta)) \\ &= \mathbf{X}^T(\mathbf{y} - F(\mathbf{X}^T\beta))\end{aligned}$$

- **Hessian**

  Here, the explicit, analytic Hessian given that $F(\mathbf{x_i}^T\beta) = P(Y_i = 1|X, \beta)$, is given by

$$\begin{aligned}\nabla^2 f(Y_i|\beta_0,\ldots,\beta_{p-1}) &= -\begin{pmatrix} \frac{\partial^2 f}{\partial\beta_0^2} & \frac{\partial^2 f}{\partial\beta_0\beta_1} & \cdots & \frac{\partial^2 f}{\partial\beta_0\beta_{p-1}} \\ \frac{\partial^2 f}{\partial\beta_1\beta_0} & \frac{\partial^2 f}{\partial\beta_1^2} & \cdots & \frac{\partial^2 f}{\partial\beta_1\beta_{p-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial\beta_{p-1}\beta_0} & \frac{\partial^2 f}{\partial\beta_{p-1}\beta_1} & \cdots & \frac{\partial^2 f}{\partial\beta_{p-1}^2} \end{pmatrix} \\ &= \sum_{i=0}^{p-1} F(\mathbf{x_i}^T\beta)(F(\mathbf{x_i}\beta)-1)\,\mathbf{x_i}\mathbf{x_i}^T = \mathbf{X}^T\, diag(F(\mathbf{x}_i^T\beta)(F(\mathbf{x}_i^T\beta)-1))\,\mathbf{X}\end{aligned}$$

Using a combination of these values, we built a Newton-Raphson (NR) algorithm to estimate the desired model parameters. The NR algorithm is the multivariate equivalent of Newton's method, a popular algorithm that successively determines better approximations to the roots/zeroes of a single-variable function. In our case, the NR algorithm maximized the previously stated log likelihood function (3). We created an iterative process to solve for our desired model estimates, $\theta_i = (\beta_0, \beta_1, \beta_2 ...)$, where:

$$\theta_i = \overbrace{\theta_{i-1}}^{\text{Prev estimate}} - \underbrace{[\nabla^2 f(\theta_{i-1})]}_{\text{Hessian}}^{-1} \underbrace{\nabla f(\theta_{i-1})}_{\text{Gradient}}$$

for

$$\theta_0, \theta_1, ..., \theta_i, ...$$

until convergence is met, where convergence of the algorithm was defined as a negligible difference between log-likelihoods of successive model estimates, with predefined tolerance of 1e-10.

We built and implemented a pathwise coordinatewise optimization algorithm which optimized model estimates one-by-one while holding all the others fixed. The algorithm minimized an objective function which was the additive inverse of the log likelihood (3). Convergence was defined as both a small euclidean distance between successive objective functions and a small euclidean distance between successive model estimates, within tolerance. Tolerance was preset to the square root of the machine's floating-point precision.

We included a LASSO constraint to perform regularization in the pathwise coordinatewise optimization algorithm, and the resulting objective function was

$$g(\beta) = -\mathcal{L}(\beta_0, ..., \beta_p) + \lambda \sum_{j=1}^{p} |\beta_j| \tag{4}$$

where $\mathcal{L}(\beta_0, ..., \beta_p)$ was the log likelihood in (3), λ was a tuning parameter, and p was the number of predictors in the model. We obtained a path of solutions for λ values between 0 and the largest estimate from the full model, and we determined the optimal λ value by 5-fold CV based on misclassification rate.

We split the data into a training set and a test set (80% and 20% of the data respectively). We compared the optimal model obtained by LASSO regularization to the full model obtained by the NR algorithm in terms of their estimated model parameters using the training set and in terms of their prediction accuracy (quantified by their misclassification rate) on the test set. All computations were performed in the R statistical software.

3 Results

There were 212 out of 569 (37%) malignant breast cancer diagnoses in this data set. Table 1 provides a descriptive summary of the predictor variables included in the full model.

Variable	Mean (SD*)
Mean radius	14.127 (3.524)
Mean texture	19.290 (4.301)
Mean smoothness	0.096 (0.014)
Mean compactness	0.104 (0.053)
Mean concavity	0.089 (0.080)
Mean concave points	0.049 (0.039)
Mean symmetry	0.181 (0.027)
Mean fractal dimension	0.063 (0.007)

Table 1: Means and standard deviations (in parentheses) of the predictors included in the full model. *SD: standard deviation

Table 2 contains the estimated model parameters for the full model based on the NR algorithm, and estimated model parameters for the optimal model based on pathwise coordinate-wise optimization with LASSO regularization and optimal $\lambda \approx 1.5$ (see figures 1 and 2 for tuning details). The full model and optimal models elicited a final test misclassification rate of approximately 9% and 11% respectively, corresponding to the random test-training split of the data described earlier.

Furthermore, we analyzed specific error type dynamics for both models, employing both subjective "cost" matrices (penalizing specific error types and rewarding specific correct classifications), along with Receiver Operator Curves (ROC) to compare True Positive and False Positive rates over a probability threshold of classification (shown in figures 5, 6). Here, Type I and Type II adverse have a higher cost for False Positives and False Negatives, respectively, with small reward for correct malignant classification and none for the converse. Utilizing the Area Under the Curve (AUC) measure, both the full model and the optimal model elicited high sensitivity and specificity over the entire probability threshold $(0, 1)$, with AUC values of 0.976 and 0.974, respectively (figure 5). Regarding "cost", we also noted that the optimal model evinced a consistently lower cost versus the full model under the Type I adverse matrix, with negligible difference between models under the Type II adverse setting, over the entire probability threshold.

4 Conclusion

We evaluated the Newton-Raphson (NR) and pathwise coordinate-wise optimization algorithms for the logistic estimation of model parameters in a prediction model for binary classification of malignant breast cancer diagnosis based on several features of breast tissue images. Generally, standard errors of breast tissue image measurements did not seem to be associated with breast cancer diagnosis (figure 3). Additionally, many predictors were strongly correlated as indicated by figures 3 and 4, and in particular mean values and largest values were almost perfectly correlated. As a result, and because of their graspable nature,

Variable	Full	Optimal
Intercept	-0.903	-0.753
Mean radius	3.610	2.625
Mean texture	1.736	1.326
Mean smoothness	0.824	0.447
Mean compactness	-0.686	0
Mean concavity	0.996	0.513
Mean concave points	2.786	2.56
Mean symmetry	0.358	0.18
Mean fractal dimension	-0.057	-0.105

Table 2: Estimated model parameters for the full and optimal models

we included only mean values in the model. The optimal LASSO-constrained model shrunk the parameter estimates relative to the full model (with the exception of the mean fractal dimension), removing mean compactness entirely. Pathwise coordinate-wise optimization was more robust[2] than the NR method in the presence of perfectly correlated predictors (mean radius, mean perimeter, and mean area), and elicited better diagnostic performance under a Type I adverse setting. While the optimal model performed as well as the full model on the test data with respect to global misclassification, further internal and external validation must take place to infer conclusive model superiority.

5 Appendix

5.1 Figures

[2]Results were not included in the manuscript

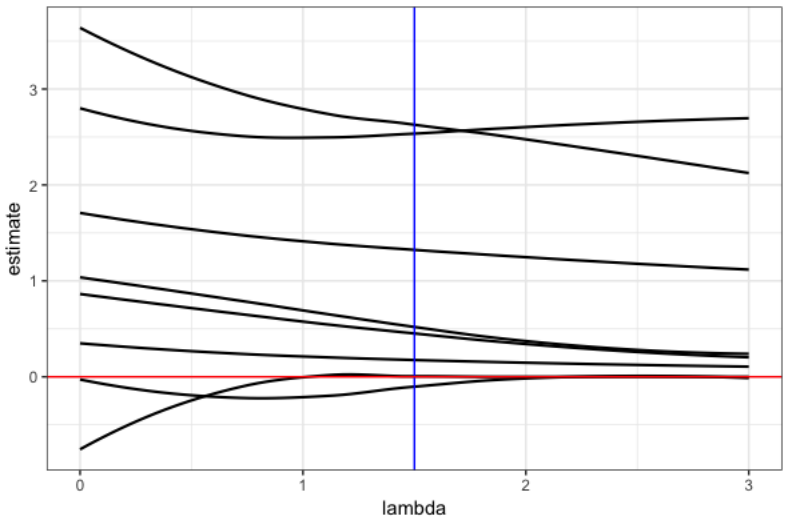

Figure 1: Solution paths for parameter estimates (excluding intercept) for a sequence of λ values. red line: $\beta = 0$, blue line: optimal $\lambda \approx 1.5$

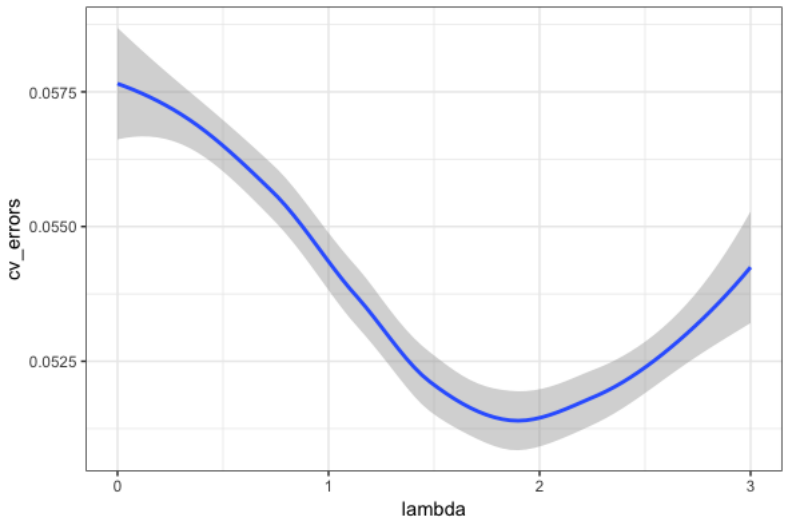

Figure 2: Training cross-validation errors for a sequence of λ values.

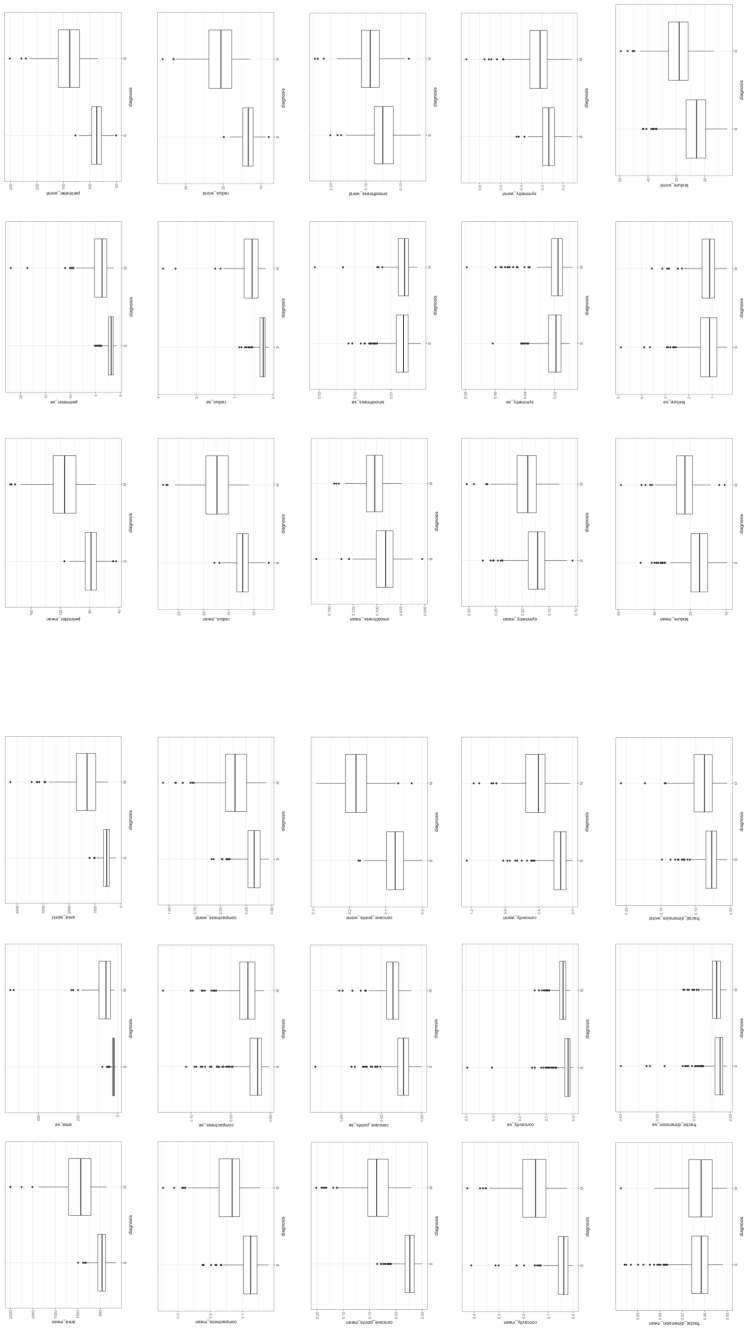

Figure 3: Box plots of the relationship between the predictors (continuous) and the outcome, breast cancer diagnosis (binary)

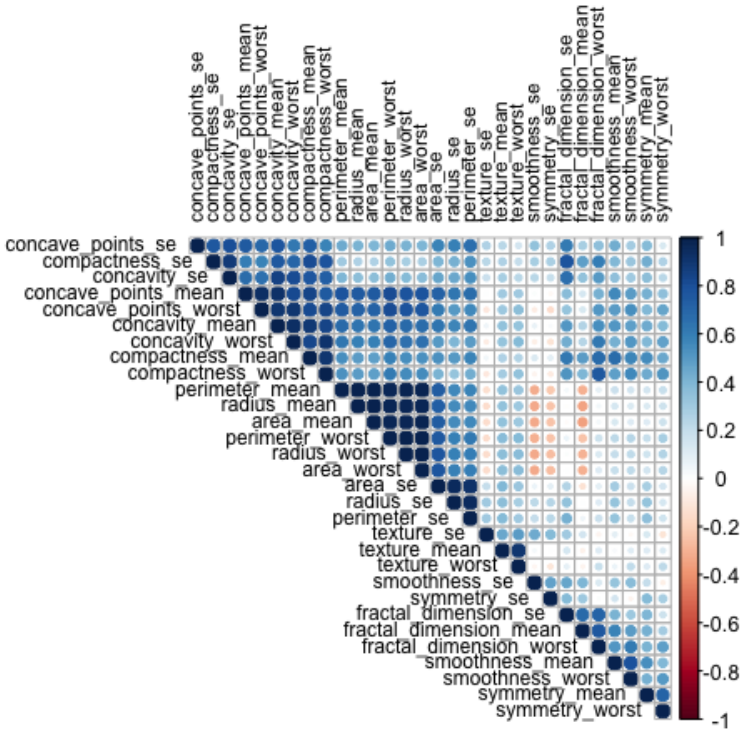

Figure 4: Correlation plot of predictors based on the Spearman's rank correlation coefficient

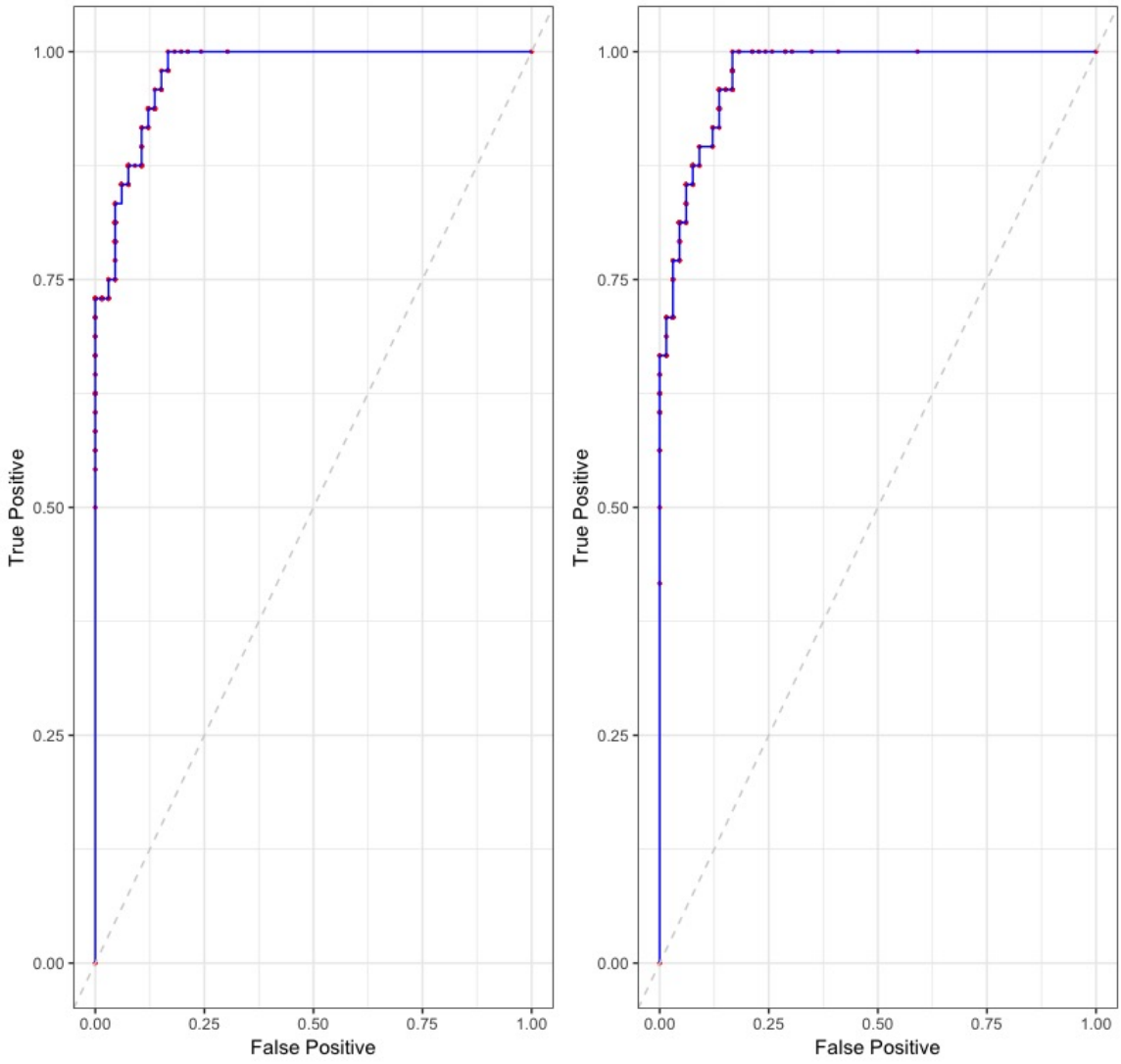

Figure 5:
Left: ROC Curve for Malignant Diagnosis with Newton-Raphson Algorithm
Right: ROC Curve for Malignant Diagnosis with LASSO constrained coordinate descent algorithm

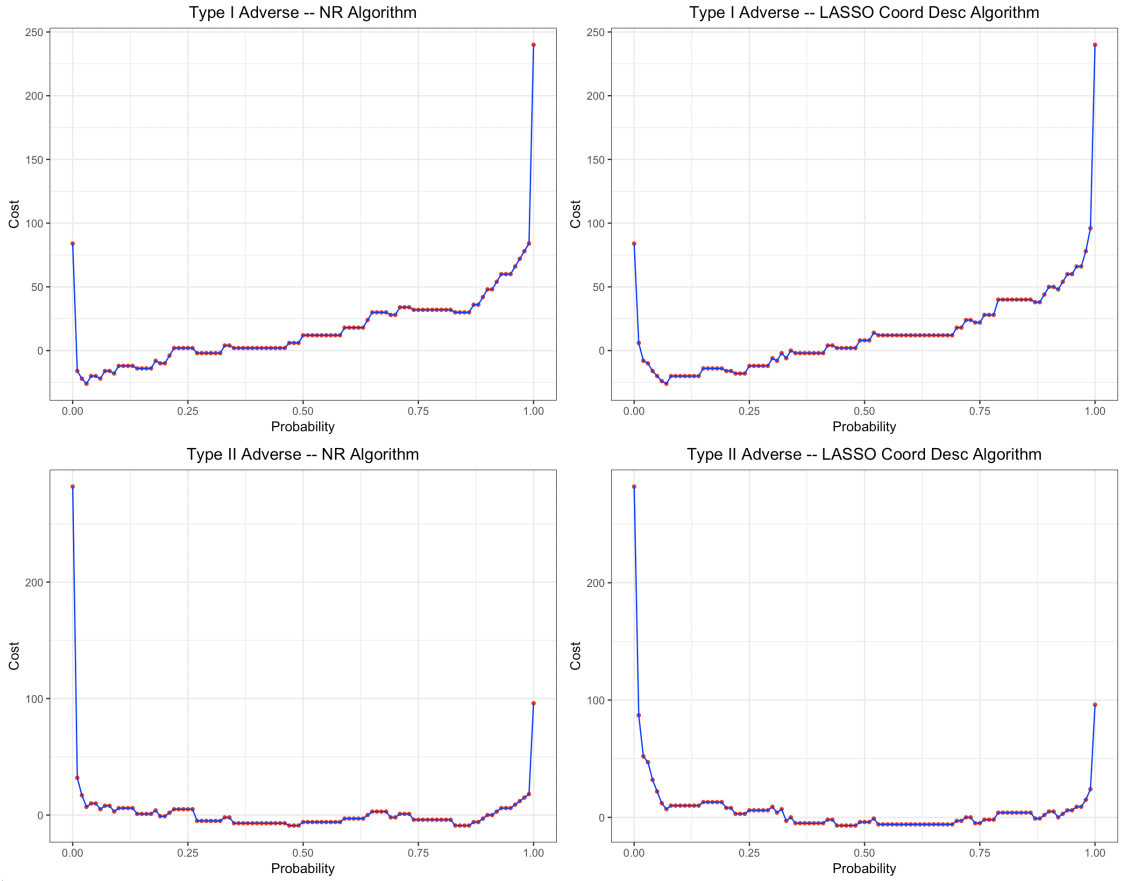

Figure 6: Classification Threshold Cost Optimization for respective algorithms and weights for types of error

6 Code

Stat_Comp Group Project 2

Muhire Kwizera (mhk2159), Quinton Neville (qn2119), Zelos Zhu (zdz2101)

March 5, 2019

Load packages/Read Data

```
#Load necessary packages
library(tidyverse)
library(readxl)
library(readr)
# library(p8105.datasets)
# library(patchwork)
library(ggridges)
library(gridExtra)
library(shiny)
library(plotly)
library(broom)
library(scales)
library(purrr)
# library(koRpus)
library(modelr)
# library(rmutil)
# library(EnvStats)
library(corrplot)
library(knitr)
library(ROCR)
library(ggplot2)

breast.df <- read_csv("./breast-cancer.csv") %>%
  janitor::clean_names() %>%
  mutate(
    diagnosis = as.factor(diagnosis) %>%
      fct_relevel("B", "M")
  )
breast.df <- breast.df[ ,-33]
```

EDA

```
# 569 observations
nobs = breast.df %>%
  nrow()

# 32 variables
nvars = breast.df %>%
  ncol()

# 1 obs per id
```

```r
nid = breast.df %>%
  pull(id) %>%
  unique()

# # summary stats
# breast.df %>%
#   select(-c(id, diagnosis)) %>%
#   summary()

# mean
meanvals = breast.df %>%
  dplyr::select(-c(id, diagnosis)) %>%
  apply(., 2, mean)

# std dev
sdvals = breast.df %>%
  dplyr::select(-c(id, diagnosis)) %>%
  apply(., 2, sd)

# descriptive stats
mean_sd = cbind(meanvals, sdvals) %>%
  as.tibble %>%
  .[1:10,] %>%
  mutate(interpretation_factor = sdvals - meanvals) %>%
  round(., 3) %>%
  mutate(variable = names(breast.df)[3:12]) %>%
  dplyr::select(variable, everything())

# mean_sd %>%
#   kable()

# Might need to center and scale (done)
# Only include mean and standard deviation in report

# # scatter plot matrix: Shows there are monotone curvilinear relatinoships.
# # Spearman correlation appropriate
# breast.df %>%
#   select(-c(id, diagnosis)) %>%
#   .[,21:30] %>%
#   pairs()

# proportion of malignant breast cancer diagnosis
prop_malign = breast.df %>%
  mutate(diagnosis = ifelse(diagnosis == "M", 1, 0)) %>%
  pull(diagnosis) %>%
  # sum()
  mean()

# Correlation plot (relationship among predictors)
breast.df %>%
  dplyr::select(-c(id, diagnosis)) %>%
  cor(method = "spearman") %>%
  corrplot(type = "upper", order = "hclust", tl.col = "black", tl.cex = 0.75)
```

```
# Most probably wil shrink about 50% of variables to 0

# Funtion for visualizing type of relationship between predictors and binary outcome
sigmoids = function(dat, var) {
  xmin = dat %>% pull(var) %>% min()
xmax = dat %>% pull(var) %>% max()

dat %>%
  mutate(diagnosis = ifelse(diagnosis == "M", 1, 0)) %>%
  dplyr::select(var, diagnosis) %>%
  ggplot(aes_string(x = var, y = "diagnosis")) +
  geom_point() + geom_smooth() + xlim(xmin, xmax) + ylim(0, 1)
}

# box plots (relationships between predictors and outcome). Many predictors seem significant
boxplots = function(dat, var) {
  dat %>%
  dplyr::select(diagnosis, var) %>%
  ggplot(aes_string(x = "diagnosis", y = var)) +
  geom_boxplot()
}

vars = names(breast.df)[-c(1,2)] %>%
  as.array()

# A priori choice of predictors (based on box plots). 15 choices:
# concave_points_worst
# concavity_worst
# compactness_worst
# area_worst
# perimeter_worst
# radius_worst
# area_se
# perimeter_se
# radius_se
# concave_points_mean
# concavity_mean
# compactness_mean
# area_mean
# perimeter_mean
# radius_mean

# Overall means and worst values seem to be more related the outcome.
# We might restrict our modelling to these (especially means).
```

Build NR Algorithm

```
mod.df <- breast.df  %>%
  dplyr::select(-c(id)) %>%
  dplyr::select(diagnosis, radius_mean, texture_mean, smoothness_mean, compactness_mean, concavity_mean
```

```r
set.seed(1)
trainidx = sample(1:nrow(mod.df), 455)
test.df <- mod.df[-trainidx, ]
train.df <- mod.df[trainidx, ]

design.matrix <- function(mod.df) {
dat <- mod.df %>%
  dplyr::select(-diagnosis) %>%
  mutate(intercept = rep(1, times = nrow(.))) %>%
  dplyr::select(intercept, everything()) %>%
  as.matrix()
dat[ ,-1] <- scale(dat[ ,-1])
y <- ifelse(mod.df$diagnosis == "M", 1, 0) #Indictor
  return(list(matrix = dat, response = y))
}


logisticstuff <- function(dat, betavec, y){
  mu <- dat %*% betavec
  p <- (exp(dat %*% betavec))/(1 + exp(dat %*% betavec))
  #Loglik, Grad, Hess
  loglik <- sum(y * mu - log(1 + exp(mu)))
  grad <- t(dat) %*% (y - p)
  Hess <- t(dat) %*% diag((p * (p - 1)) %>% as.vector()) %*% dat %>% forceSymmetric() %>% as.matrix()
  return(list(loglik = loglik, grad = grad, Hess = Hess))
}


NR_logit <- function(mod.df, tol = 1e-10, maxiter = 100){
  #Scaled design matrix
  dat <- design.matrix(mod.df)$matrix
  y <- design.matrix(mod.df)$response
  #Initialize
  i <- 0
  #Initial beta vector guess
  current <- runif(ncol(dat), 0, 1)
  equations <- logisticstuff(dat, current, y)
  res <- c(0, equations$loglik, current)
  prevloglik <- -Inf
  #Loop until convergence

  while (i < maxiter & abs(equations$loglik - prevloglik) > tol) {
    i <- i + 1
    prevloglik <- equations$loglik
    prev <- current
 #  current <- solve(equations$Hess, crossprod(equations$Hess, prev) - equations$grad)  #bugs out somet
    current <- prev - solve(equations$Hess) %*% equations$grad
    equations <- logisticstuff(dat, current, y) # log-lik, gradient, Hessian

    res <- rbind(res, c(i, equations$loglik, current))
    # Add current values to results matrix
  }
  beta.final <- res[nrow(res), -c(1, 2)] %>% as.matrix() #pull final coef, remove loglik/iter
```

```
  return(list(result = res, beta.vec = beta.final))
}

NR_logit(train.df)

#Compare to glm output
train.glm.df <- train.df[ ,-1] %>% scale() %>% as.data.frame() %>%
  mutate(diagnosis = train.df$diagnosis)
glm(diagnosis ~ ., family = binomial(link = "logit"),
    data = train.glm.df) %>%
  broom::tidy() %>% knitr::kable(digits = 5)
```

Coordinate Descent Algorithm

```
full_vars = names(breast.df) %>%
  .[-c(1,2)] %>%
  head(., 10) %>%
  .[-c(3, 4)]
dat <- breast.df  %>%
  dplyr::select(-c(id, diagnosis)) %>%
  mutate(intercept = rep(1, nrow(.))) %>%
  dplyr::select(intercept, full_vars) %>% ##radius_mean, texture_mean, smoothness_mean, compactness_mean
  as.matrix()

dat[,-1] <- scale(dat[,-1])

y <- ifelse(breast.df$diagnosis == "M", 1, 0) #Indictor

# Splitting data into training set and test set (80/20 split)
testX = dat[-trainidx,]
testY = y[-trainidx]
dat = dat[trainidx,]
y = y[trainidx]


log.out <- NR_logit(train.df)
lambda_max = log.out$beta.vec %>%
  abs() %>%
  max() %>%
  floor()

# analysis of probability of having malignant breast cancer
# unit increase in standardized varaible corresponds to increase in x by (sd - mean)

# Logistic with LASSO minimization function
log_lasso = function(betavec_now, x_now,
                     betavec_other, x_other,
                     beta_mult_now, beta_mult_other,
                     y, lambda = 0) {

  betavec_now = betavec_now %>% as.array()
```

```
  betavec_other = betavec_other %>% as.array()

  x_now = x_now %>% as.matrix()
  x_other = x_other %>% as.matrix()

  y = y

  mu <- x_now %*% betavec_now + x_other %*% betavec_other

  # Maximizing log lik is same as minimizing neg log lik
  loglik_neg <- -sum(y*mu - log(1 + exp(mu)))

  penalty = lambda * (abs(betavec_now * beta_mult_now) + sum(abs(betavec_other %*% beta_mult_other)))

  obj_fctn = loglik_neg + penalty

  return(obj_fctn)
}

# Optimize function in 1D
optimize1D = function(idx, dat, betavec, y, lambda = 0) {

  absbeta_multpr = c(0, rep(1, (length(betavec) - 1)))

  # Minimize objective function in 1-D while fixing everything else. Search interval is [-20, 20]
  min_obj = optimize(log_lasso,
                     -20:20, dat[,idx],
                     betavec[-idx], dat[,-idx],
                     absbeta_multpr[idx], absbeta_multpr[-idx],
                     y, lambda)

  return(list(idx = idx, minimum = min_obj$minimum, objective = min_obj$objective))
}

# function for updating betavec after each optimization
update_betas = function(idx, dat, betavec, y, lambda = 0) {
  min_obj = optimize1D(idx, dat, betavec, y, lambda)
  betavec[min_obj$idx] = min_obj$minimum

  return(list(betavec = betavec, objective = min_obj$objective))
}

coord_desc = function(dat, y, betavec, lambda = 0) {
  #-------------------Initialize-------------------------------
  tol = sqrt(.Machine$double.eps)
differ_obj = 1
differ_beta = 1
mod = ncol(dat) + 1

idx = 0

obj_min = 0
solpath = NULL
```

```r
# loop through till convergence
while (differ_obj > tol | differ_beta > tol) {
  idx = max(c(1, (idx + 1) %% mod)) # increment index

  old_betavec = betavec
  old_obj_min = obj_min
  solpath = rbind(solpath, betavec)

  # print(betavec)
  # print(obj_min)

  updates = update_betas(idx, dat, betavec, y, lambda)

  betavec = updates$betavec
  obj_min = updates$objective

  differ_obj = abs(obj_min - old_obj_min)
  differ_beta = sum((betavec - old_betavec)^2)
}

# Solution for current lambda value
sol_curr_lambda = tail(solpath, 1) %>% as.array()

return(sol_curr_lambda)

}

# coord_desc(dat, y = y, betavec = betavec , 0)

# # Fit coordinate descent for multiple lambda values
lambdas = seq(from = 0, to = lambda_max, length.out = 70) %>% as.array()

set.seed(1)
betavec <- runif(ncol(dat), 0, 1)
solutions = apply(lambdas, 1, FUN = coord_desc, dat = dat, y = y, betavec = betavec)
```

Cross-validation to select lambda

```r
# Compute misclassification rate for each lambda value
misscl_rate = function(dattest, solutions, s = 1,  ytest) {

  #  # ***********************************************************
  # # this code was used to determine optimal probability cutoff
  #  # for predicting binary outcomes.
  #  # 0.5 was decent so kept it because it's fair
  #  # predict y
  # mupred = dat %*% solutions[, s]
  # probpred = exp(mupred) / (1 + exp(mupred))
  #
  # # get performance measures
  # pred.cd = prediction(mupred, y)
```

```
  # perf.cd = performance(pred.cd,"tpr","fpr")
  # plot(perf.cd)
  #
  # # looking at sensitivity and 1 - specificity for each cutoff
  # fpr_tpr_tbl = tibble(cutoff = perf.cd@alpha.values %>% unlist(),
  #                      fpr = perf.cd@x.values %>% unlist,
  #                      tpr = perf.cd@y.values %>% unlist)
  # # ****************************************************************

  # predict y
  mupred = dattest %*% solutions[, s]
  probpred = exp(mupred) / (1 + exp(mupred))
  ypred = ifelse(probpred > 0.5, 1, 0)

  # missclassification rate
  misscl = mean(ypred != ytest)

  return(misscl)
}

# implementing the heavy lifting of cross validation
implement_cv = function(folds, fold, y, dat, lambdas, betavec) {
  # split data into training and testing sets
ytest = y[folds == fold]
dattest = dat[folds == fold,]
ytrain = y[folds != fold]
dattrain = dat[folds != fold,]

# implement coordinate descent algorithm for all lambda choices
solutions = apply(lambdas, 1, FUN = coord_desc, dat = dattrain, y = ytrain,  betavec = betavec)
msclass_rate = apply(1:ncol(solutions) %>% as.array, 1, FUN = misscl_rate,
      solutions = solutions, dattest = dattest, ytest = ytest)

return(msclass_rate)
}

# cross validation function
cv_fctn = function(k = 5, y, dat, lambdas, betavec) {
  # cross validation
  # k = 5
  set.seed(1)
  folds = sample(1:k, nrow(y %>% as.array()), replace = TRUE)

  mscl_rates = apply(
  1:k %>% as.array,
  1,
  FUN = implement_cv,
  folds = folds,
  y = y,
  dat = dat,
  lambdas = lambdas,
  betavec = betavec
  )
```

```
  cv_errors = apply(mscl_rates, 1, mean)

  best.lambda = lambdas[which.min(cv_errors)]

  # Plot cross validation errors
  cvplot = tibble(lambda = lambdas,
  cv_error = cv_errors) %>%
  ggplot(aes(x = lambda, y = cv_errors)) + geom_smooth()

  return(list(best.lambda = best.lambda, cvplot = cvplot))
}

# cross validation
cv.out = cv_fctn(k = 5, y, dat, lambdas, betavec)

#Optimal model using
optimal_sol = coord_desc(dat, y = y, betavec = betavec , cv.out$best.lambda) %>%
  round(., 3)
optimal_sol
```

Evaluating performance/Predicting outcome on test data set based on full and optimal models

```
# function for predicting outcome and getting misclassification error
predicty = function(testX, betavec, testY) {
  # predict y
  mupred = testX %*% betavec
  probpred = exp(mupred) / (1 + exp(mupred))
  ypred = ifelse(probpred > 0.5, 1, 0)

  # misclassification rate
  misscl = mean(ypred != testY)

  return(list(missclassification = misscl,
              probs= probpred))
}

# misclassification rates for newton's and cd
predicty(testX, log.out$beta.vec, testY)$missclassification
predicty(testX, (optimal_sol %>% as.vector), testY)$missclassification
```

Appendix

Figure 1

```
# Illustrating LASSO shrinkage
solutions %>%
  t() %>%
```

```
  .[,-1] %>%
  as.tibble() %>%
  mutate(lambda = lambdas) %>%
  # mutate(variable = replace(variable, c(paste0("V", 1:(ncol(dat) - 1))), full_vars)) %>%
  gather(variable, estimate, paste0("V", 1:(ncol(dat) - 1))) %>%
  # mutate(variable = replace(variable, c(paste0("V", 1:(ncol(dat) - 1))), full_vars)) %>%
  ggplot(aes(x = lambda, y = estimate, group = variable)) +
  geom_smooth(se = FALSE, color = "black", size = 0.75) +
  geom_vline(color = "blue", xintercept = 1.5, show.legend = TRUE) +
  geom_hline(color = "red",yintercept = 0)
```

Figure 2

```
cv.out$cvplot
```

Figure 3/4

```
#Refer to data_eda chunk
bxplts

breast.df %>%
  dplyr::select(-c(id, diagnosis)) %>%
  cor(method = "spearman") %>%
  corrplot(type = "upper", order = "hclust", tl.col = "black", tl.cex = 0.75)
```

Figure 5 (Roc Curves)

```
multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

 if (numPlots==1) {
    print(plots[[1]])

  } else {
    # Set up the page
```

```
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                      layout.pos.col = matchidx$col))
    }
  }
}
```

```
#Split into test and train
test.df <- mod.df[-trainidx, ]
train.df <- mod.df[trainidx, ]
prob <- function(vec) {
  exp(vec) / (1 + exp(vec))
}
predict.NR.glm <- function(model, new.data, p.threshold = 0.5) {
  test.mat <- design.matrix(new.data)$matrix
  response <- design.matrix(new.data)$response
  #Probability and Misclassification
  probs <- test.mat %*% model$beta.vec %>% prob()
  preds <- ifelse(probs > p.threshold, 1, 0)
  misclassification <- ifelse(preds == response, 0, 1) %>% mean()
  return(list(probs = probs, preds = preds, error = misclassification))
}
#################ROC AUC Functions#################################
Roc.Log <- function(result,y.pred.r0){
  probs <- seq(0,1,by=0.005)
  roc.log <- matrix(0,nrow=length(probs),ncol=2)
  result <- ifelse(result == "M", TRUE, FALSE)
  i <- 1
  for(p in probs){
    pred <- y.pred.r0 > p
    ##False positive rate
    false.pos <- with(test.df,
                      sum(!result & pred)/sum(!result))
    ##True postive rate
    true.pos <- with(test.df,
                     sum(result & pred)/sum(result))
    roc.log[i,] <- c(false.pos,true.pos)
    i <- i+1
  }
  return(roc.log)
}

plotRoc <- function(roc.log,charstring){
  data.frame(FP =rev(roc.log[,1]),TP=rev(roc.log[,2])) %>%
    ggplot()+
    geom_point(aes(FP,TP),color="red",size=.5) +
    geom_step(aes(FP,TP),color="blue") +
```

```
    geom_abline(slope = 1, linetype = 2, colour = "lightgrey") +
    labs(
      x = "False Positive",
      y = "True Positive"
      title = paste0("ROC Curve for Malignant Diagnosis with " ,charstring)
    )
}
auc <- function(roc){
  len <- nrow(roc)
  ##The "delta X" values
  delta <- roc[-1,1]-roc[-len,1]
  ##The "heights" the rectangle (drop the first or last).
  hgt <- roc[-1,2]
  ##The Riemann Sum
  sum(-delta*hgt)
}

fit <- NR_logit(train.df)

preds <- predict.NR.glm(fit, test.df)$prob
roc.log <- Roc.Log(test.df$diagnosis, preds)
auc(roc.log)

preds.lasso <- predicty(testX, (optimal_sol %>% as.vector), testY)$probs
roc.log.lasso <- Roc.Log(ifelse(testY == 1, "M", "B"), preds.lasso)
auc(roc.log.lasso)

multiplot(plotRoc(roc.log, " "), plotRoc(roc.log.lasso, " "), cols =2)
```

Figure 6 Cost Analysis

```
cost_type1 <- function(t, p){
    if(t & !p)
        return(5) #False positive weight
    if(!t & p)
        return(2) #False Negative
  if(t & p)
    return(-1) #True Positive prediction
    return(0)  #True Negative
}
cost_type2 <- function(t, p){
    if(t & !p)
        return(2) #False positive weight
    if(!t & p)
        return(5) #False Negative
  if(t & p)
    return(-1) #True Positive prediction
    return(0)  #True Negative
}

p <- seq(0, 1, by = 0.01)
fit_NR <- NR_logit(train.df)
```

```r
preds_NR <- predict.NR.glm(fit_NR, test.df)$probs #grab predicted probs
result_NR <- ifelse(design.matrix(test.df)$response == 1, TRUE, FALSE)

preds_lasso <- predicty(testX, (optimal_sol %>% as.vector), testY)$probs
result_lasso <- ifelse(testY == 1, TRUE, FALSE)

cost_vec_NR_type1 <- c()
cost_vec_NR_type2 <- c()
cost_vec_lasso_type1 <- c()
cost_vec_lasso_type2 <- c()

#Optimize threshold test data
for (i in 1:length(p)) {
    pred_NR <- preds_NR > p[i]
    frame_NR <- with(test.df, cbind(result_NR, pred_NR))
    cost_vec_NR_type1 <- c(cost_vec_NR_type1, sum(apply(frame_NR, 1, function(ls) cost_type1(ls[1], ls[
    cost_vec_NR_type2 <- c(cost_vec_NR_type2, sum(apply(frame_NR, 1, function(ls) cost_type2(ls[1], ls[


    pred_lasso <- preds_lasso > p[i]
    frame_lasso <- with(test.df, cbind(result_lasso, pred_lasso))
    cost_vec_lasso_type1 <- c(cost_vec_lasso_type1, sum(apply(frame_lasso, 1, function(ls) cost_type1(ls
    cost_vec_lasso_type2 <- c(cost_vec_lasso_type2, sum(apply(frame_lasso, 1, function(ls) cost_type2(ls
  }

#Visualize

#Type 1 Adverse - NR
type1_NR<- tibble(
  probability = p,
  cost = cost_vec_NR_type1
) %>%
  ggplot() +
    geom_point(aes(probability, cost), color = "red", size= 1) +
    geom_line(aes(probability, cost), color = "blue") +
  labs(
    y = "Cost",
    x = "Probability",
    title = "Type I Adverse -- NR Algorithm"
  )
opt_p_cost_NR_type1 <- cbind(p, cost_vec_NR_type1)[which.min(cost_vec_NR_type1), 1]

#Type 2 Adverse - NR
type2_NR<- tibble(
  probability = p,
  cost = cost_vec_NR_type2
) %>%
  ggplot() +
    geom_point(aes(probability, cost), color = "red", size= 1) +
    geom_line(aes(probability, cost), color = "blue") +
  labs(
    y = "Cost",
    x = "Probability",
```

```
    title = "Type II Adverse -- NR Algorithm"
  )
opt_p_cost_NR_type2 <- cbind(p, cost_vec_NR_type2)[which.min(cost_vec_NR_type2), 1]


#Type 1 Adverse - lasso
type1_lasso <- tibble(
  probability = p,
  cost = cost_vec_lasso_type1
) %>%
  ggplot() +
    geom_point(aes(probability, cost), color = "red", size= 1) +
    geom_line(aes(probability, cost), color = "blue") +
  labs(
    y = "Cost",
    x = "Probability",
    title = "Type I Adverse -- LASSO Coord Desc Algorithm"
  )
opt_p_cost_lasso_type1 <- cbind(p, cost_vec_lasso_type1)[which.min(cost_vec_lasso_type1), 1]

#Type 2 Adverse - lasso
type2_lasso <- tibble(
  probability = p,
  cost = cost_vec_lasso_type2
) %>%
  ggplot() +
    geom_point(aes(probability, cost), color = "red", size= 1) +
    geom_line(aes(probability, cost), color = "blue") +
  labs(
    y = "Cost",
    x = "Probability",
    title = "Type II Adverse -- LASSO Coord Desc Algorithm"
  )
opt_p_cost_lasso_type2 <- cbind(p, cost_vec_lasso_type2)[which.min(cost_vec_lasso_type2), 1]


multiplot(type1_NR, type2_NR, type1_lasso, type2_lasso, cols = 2)

opt_p_cost_NR_type1
opt_p_cost_NR_type2
opt_p_cost_lasso_type1
opt_p_cost_lasso_type2
```