Machine Learning Homework 3

Latera Tesfaye Olana

20 February, 2023

Question 1

Question 1a: The following table provides summary of the given data.

Table 1: Summary table of the data

diagnosis	observations
Benign	357
Malignant	212

The given data has 357 Benign outcome observations and 212 Malignant observations. The data has two predictors (p=30). Overall, the data has 569 observations. This data has no missing values.

Question 1b:
The data is divided into two, training and test with each containing 70.3 and 29.7, fo the data respectively.

Question 1c:

It is imperative to perform normalization separately on the training and test sets to prevent data leakage. If the entire dataset (i.e., both training and test) is normalized before splitting, the training set is normalized using information from the test set, leading to overfitting.

It is essential to normalize the training and test sets separately, using only information from their respective sets. This approach ensures that the normalization is not based on information from the other set, which helps to ensure that the model can generalize well to unseen data.

Question 1d:

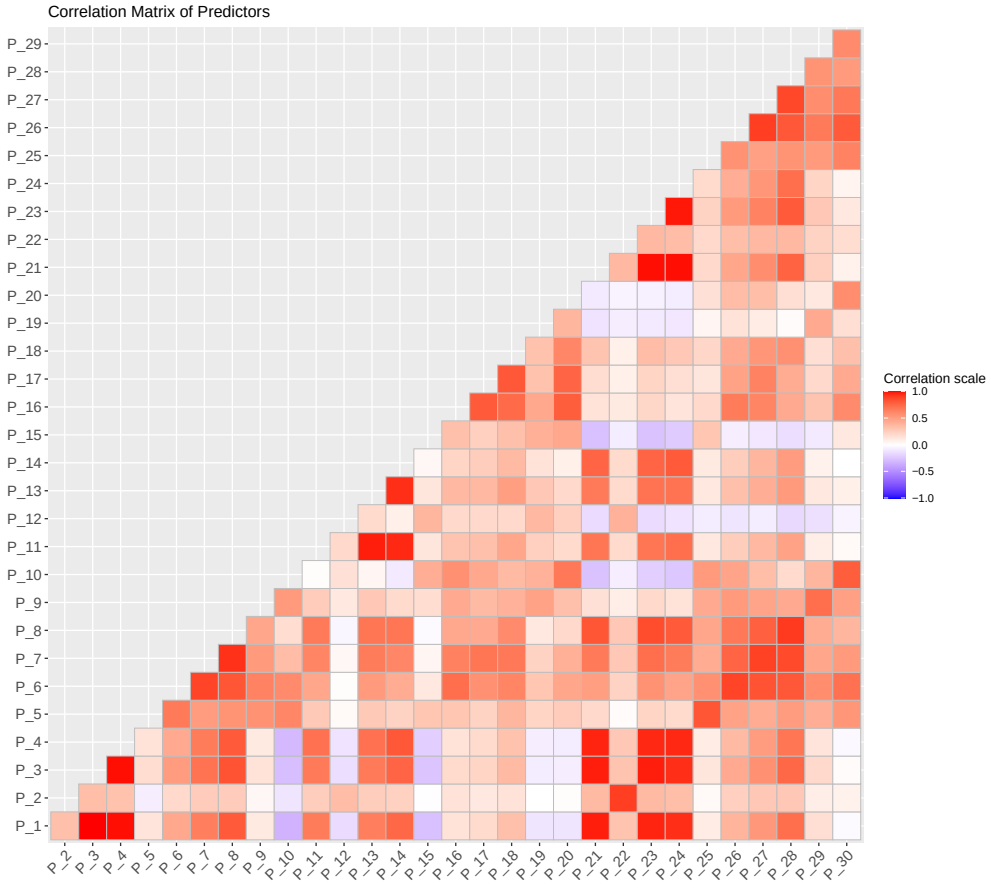

Figure 1: Correlation matrix for our train data

As it is shown in figure 1, some of the predictors are strongly correlated (both directions). This might have the following *general* complecations (cited):

Overfitting: High correlation between predictors can lead to overfitting in a classification model. When two or more predictors are highly correlated, they can carry redundant information, causing the model to give more importance to these predictors than necessary. This can lead to a model that performs well on the training data but poorly on the test data.

Model Interpretability: Correlated predictors can make it difficult to interpret the model results. When two or more predictors are highly correlated, it can be difficult to determine which predictor is actually driving the classification decision. (Lack of interpretability in most of ML is models, kind of make us to not worry that much about this issue)

Running time Complexity: High dimensional data with correlated predictors can increase the computational complexity of a classification model. This is because the model needs to consider all possible combinations of predictors to determine which ones are most important.

Bias: Correlation between predictors can lead to bias in a classification model. If two predictors are highly correlated, and one of them is more important for the classification decision, the model may assign too much weight to the correlated predictor, leading to a biased model.

Question 1e:

After fitting simple logistic regression, Table 2 shows the fitted coefficients for each predictors.

Table 2: Logistic regression training coefficients

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	92.46	28281.57	0.00	1.00
P_1	-863.00	462270.50	0.00	1.00
P_2	-3.80	13416.45	0.00	1.00
P_3	989.13	544553.10	0.00	1.00
P_4	-189.77	100538.36	0.00	1.00
P_5	117.78	15574.74	0.01	0.99
P_6	-302.08	41791.18	-0.01	0.99
P_7	-8.09	39076.94	0.00	1.00
P_8	265.44	49259.17	0.01	1.00
P_9	-122.56	22339.35	-0.01	1.00
P_10	96.25	28237.70	0.00	1.00
P_11	559.97	61288.54	0.01	0.99
P_12	3.35	9628.51	0.00	1.00
P_13	-110.48	70063.53	0.00	1.00
P_14	-540.24	107218.22	-0.01	1.00
P_15	31.71	12943.54	0.00	1.00
P_16	73.75	13649.61	0.01	1.00
P_17	52.34	57193.23	0.00	1.00
P_18	90.04	20354.30	0.00	1.00
P_19	-93.14	24992.86	0.00	1.00
P_20	-262.42	65894.49	0.00	1.00
P_21	-1030.88	195279.90	-0.01	1.00
P_22	105.94	15834.64	0.01	0.99
P_23	-502.59	175318.35	0.00	1.00
P_24	2156.15	279421.09	0.01	0.99
P_25	-97.37	20531.90	0.00	1.00
P_26	5.47	30882.30	0.00	1.00
P_27	106.24	52714.25	0.00	1.00

	Estimate	Std. Error	z value	Pr(>|z|)
P_28	162.49	25575.21	0.01	0.99
P_29	241.66	25846.59	0.01	0.99
P_30	-39.36	62867.79	0.00	1.00

The correlation between the variable *P_3* and *P_1* is -0.98. Meaning they are highly correlated. The coefficient estimates are, $\hat{\beta}_1$ = -863 and $\hat{\beta}_3$ = 989.13. These two $\hat{\beta}$ values are large (as compared to other non-correlated predictors) and opposite to each other. Accordingly, as stated above (in question 1d) inclusion of correlated predictors in many ways, will generate incorrect results from our models.

Question 1f:

Table 3: Confussion matrix for train data

	Actual Benign	Actual Malignant
Benign	255	0
Malignant	0	145

Table 4: Confussion matrix for train data

	Percentages
Accuracy	1.0000
Sensitivity	1.0000
Specificity	1.0000
Pos Pred Value	1.0000
Neg Pred Value	1.0000
Prevalence	0.3625
Detection Rate	0.3625

As shown in table 3 and 4 the accuracy, sensitivity, and specificity of the model were estimated to be 100% (no miss classification). However, the detection rate and prevalence were estimated to be 36.25%. The model is good at correctly identifying negative cases but struggles with identifying positive cases due to the low number of positive cases in the dataset (one possible reason). This can be a common issue in imbalanced datasets where the positive class is rare.

Table 5: Confussion matrix for test data

	Actual Benign	Actual Malignant
Benign	99	7
Malignant	3	60

Table 6: Confussion matrix for test data

	Percentages
Accuracy	0.94083
Sensitivity	0.89552
Specificity	0.97059
Pos Pred Value	0.95238

	Percentages
Neg Pred Value	0.93396
Prevalence	0.39645
Detection Rate	0.35503

As shown in table 5 and 6 the test accuracy is 94.0828402%. The model's sensitivity and specificity decreased as well. This is a classical sign of over-fitting.

Question 2

Question 2a:

The data is converted into matrix. The new sets are: *X_study_trn, X_study_tst* for predictors train and test, whereas, *Y_study_trn, Y_study_tst* for outcome train and test.

Question 2b:

We fitted glmnet (using a family of *binomial*) using the given different λ values.

Question 2c:

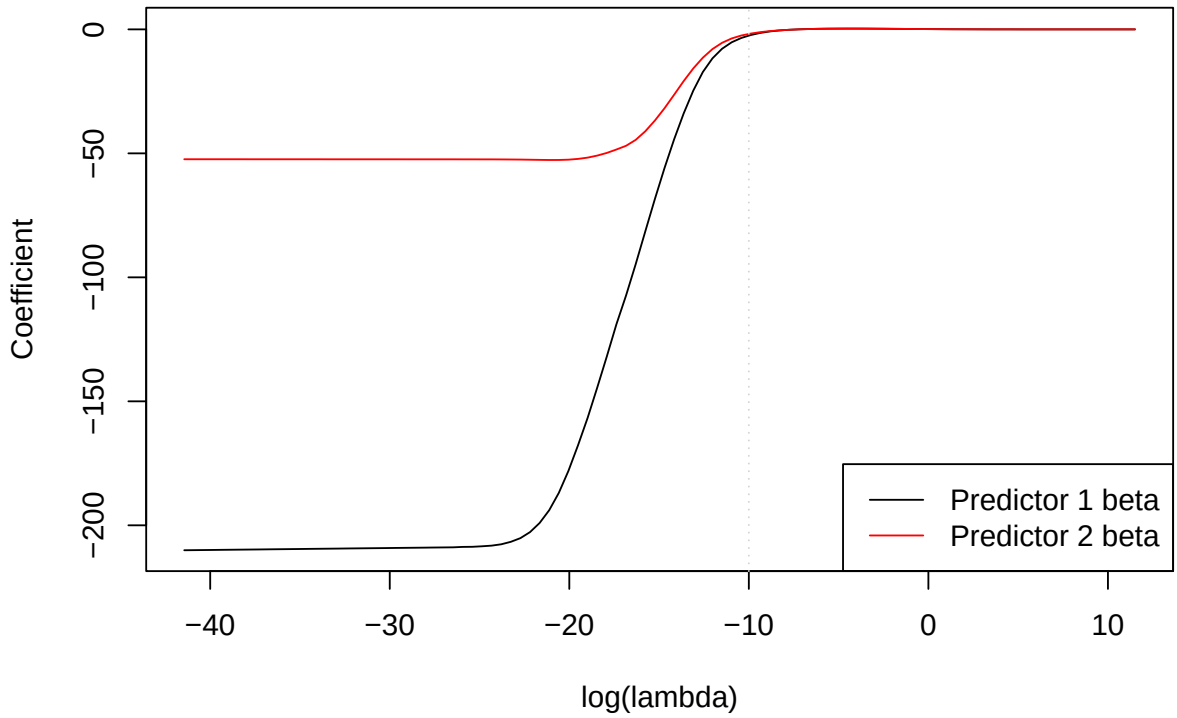

Figure 2: log(lambda) Vs beta values for predictor 1 and predictor 3 - ridge fit

As sown in figure 2, as the value of λ increases the estimated coefficient for the first predictor (P_1) is penalized more (hence the rapid drop). Before both coefficients started diverging, around λ greater than -10, both β values where close to zero.

Question 2d:

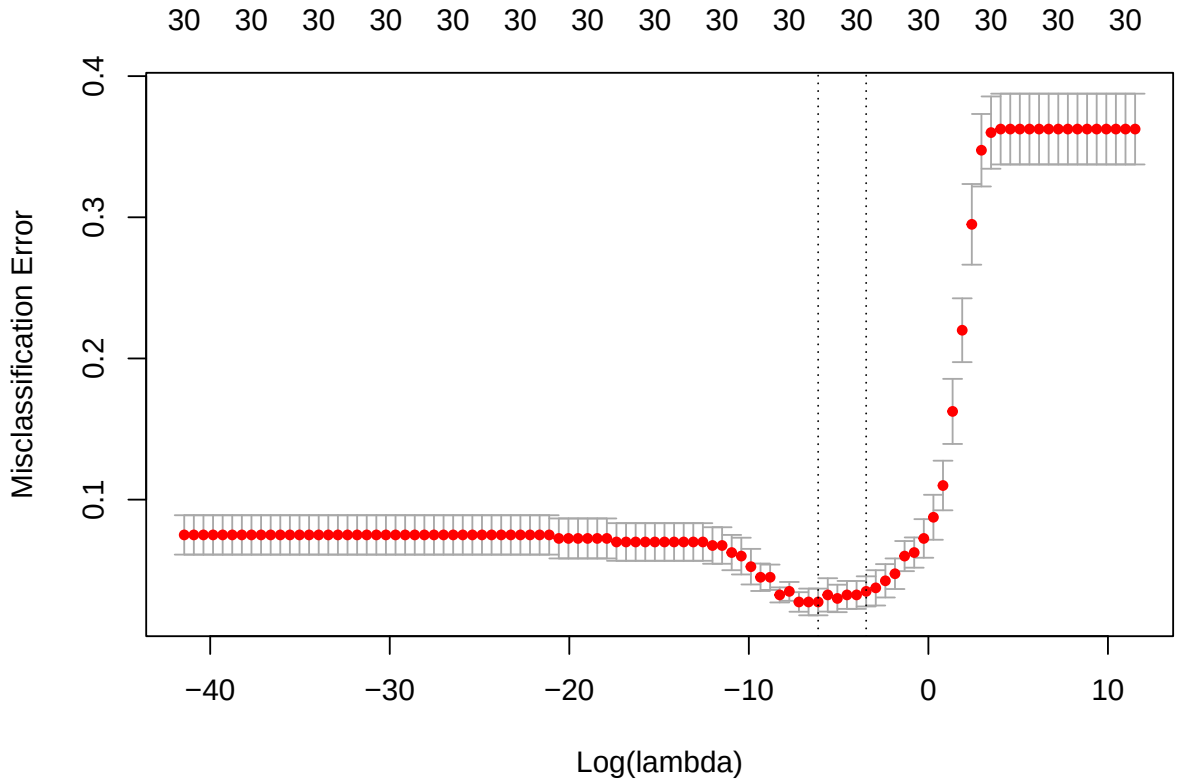

Figure 3: Misclassification Error vs. Log(lambda)

The minimum selected value of $\lambda = 0.00215$. Table 7 also shows the extracted coefficients ($\hat{\beta}$) of the selected model.

Table 7: Coefficients of the selected model, with CV

	beta values
(Intercept)	-0.54698
P_1	0.22386
P_3	0.20805
P_4	0.36057
P_5	0.08255
P_6	-0.57082
P_7	0.72063
P_8	0.82068
P_9	-0.14163
P_10	-0.21189
P_11	1.32206
P_12	-0.29797
P_13	0.65887
P_14	1.09874
P_15	0.24043

	beta values
P_16	-0.68354
P_17	0.12515
P_18	0.50646
P_19	-0.34902
P_20	-0.86524
P_21	0.82465
P_22	1.17312
P_23	0.57898
P_24	0.93166
P_25	0.77784
P_26	-0.12351
P_27	1.14174
P_28	0.95771
P_29	1.05582
P_30	0.17492

Question 2e:

As shown in table 7,there are no zero coefficients (i.e, all β values for all the predictors are non zero). A coefficient close to zero is that of predictor 9 (-0.14163). These are the coefficients that correspond to features that have little or no impact on the target variable and can be considered irrelevant for the model. After performing cross-validation to determine the optimal regularization parameter (lambda), the resulting beta values will typically be non-zero for all variables in the model, but the magnitude of these coefficients will vary depending on the value of λ. As λ increases, the magnitude of the coefficients will decrease and approach zero. Eventually, some coefficients may become exactly zero as λ becomes very increases.

Question 2f:

Table 8: Confussion matrix for train data

	Actual Benign	Actual Malignant
Benign	255	0
Malignant	3	142

Table 9: Confussion matrix for train data

	Percentages
Accuracy	0.99250
Sensitivity	1.00000
Specificity	0.98837
Pos Pred Value	0.97931
Neg Pred Value	1.00000
Prevalence	0.35500
Detection Rate	0.35500

As shown in table 8 and 9, the train data accuracy is 99.25%, sensitivity 100% and specificity 98.8372093%. Looking at the confusion matrix for train dataset, the true positive for classification was 255. The false negative is 3; the true negative 142 and finally the false positive is 0.

Table 10: Confussion matrix for test data

	Actual Benign	Actual Malignant
Benign	101	1
Malignant	2	65

Table 11: Confussion matrix for test data

	Percentages
Accuracy	0.98225
Sensitivity	0.98485
Specificity	0.98058
Pos Pred Value	0.97015
Neg Pred Value	0.99020
Prevalence	0.39053
Detection Rate	0.38462

As shown in table 10 and 11, the test data accuracy is 98.2248521%, sensitivity is 98.4848485% and specificity 98.0582524%. Looking at the confusion matrix for test dataset, the true positive for classification was 101. The false negative is 2; the true negative 65 and finally the false positive is 1.

The accuracy of the test and train set are close to each other, implying the model is performing well interms of not overfitting.

Question 2g:

Figure 3 shows the RoC plot for regularized *glm* fit.

Question 2h:

The area under the curve from the RoC curve is 99.8%.

Question 3

Question 3b:

We fitted regularized lasso (using a family of *binomial*) using the given different λ values.

Question 3c:

For lasso regression the β coefficient for predictor number three (P_3) is zero. For β estimates of predictor one (P_1), as the penalty increases, the coefficient is gradually shrinking towards zero (more radically after $\log(\lambda) =$ - 20). It reaches the value of $\log(\lambda)$ around -8 (as shown in figure 4).

Question 3d:

The minimum selected value of $\lambda = 0.00215$. Table 12 also shows the extracted coefficients ($\hat{\beta}$) of the selected model.

Table 12: Coefficients of the selected model, with CV

	beta values
(Intercept)	-0.17051
P_1	0.00000

	beta values
P_3	0.00000
P_4	0.00000
P_5	0.00000
P_6	0.00000
P_7	0.00000
P_8	0.80000
P_9	0.00000
P_10	-0.08050
P_11	2.75946
P_12	-0.39346
P_13	0.00000
P_14	0.00000
P_15	0.45332
P_16	-0.87317
P_17	0.00000
P_18	0.18516
P_19	0.00000
P_20	-0.59124
P_21	0.00000
P_22	1.52214
P_23	0.00000
P_24	3.84296
P_25	0.43381
P_26	0.00000
P_27	1.60292
P_28	1.20495
P_29	0.58450
P_30	0.00000

Question 3e:

After cross validation for lasso regularization, there are 15 β values which are zero and 15 non-zero coefficients. This is large compared to ridge regularization (but expected!).

Question 3f:

Table 13: Confussion matrix for train data

	Actual Benign	Actual Malignant
Benign	254	1
Malignant	3	142

Table 14: Confussion matrix for train data

	Percentages
Accuracy	0.99000
Sensitivity	0.99301
Specificity	0.98833
Pos Pred Value	0.97931
Neg Pred Value	0.99608

	Percentages
Prevalence	0.35750
Detection Rate	0.35500

As shown in table 13 and 14, the train data accuracy is 99%, sensitivity 99.3006993% and specificity 98.8326848%. Looking at the confusion matrix for train dataset, the true positive for classification was 254. The false negative is 3; the true negative 142 and finally the false positive is 1.

Table 15: Confussion matrix for train data

	Actual Benign	Actual Malignant
Benign	101	1
Malignant	2	65

Table 16: Confussion matrix for train data

	Percentages
Accuracy	0.98225
Sensitivity	0.98485
Specificity	0.98058
Pos Pred Value	0.97015
Neg Pred Value	0.99020
Prevalence	0.39053
Detection Rate	0.38462

As shown in table 15 and 16, the test data accuracy is 98.2248521%, sensitivity 98.4848485% and specificity 98.0582524%. Looking at the confusion matrix for test dataset, the true positive for classification was 101. The false negative is 2; the true negative 65 and finally the false positive is 1.

Question 3g:

Figure 5 shows the RoC for lasso fit.

Question 3h:

The area under the curve from the RoC curve is 99.8%.

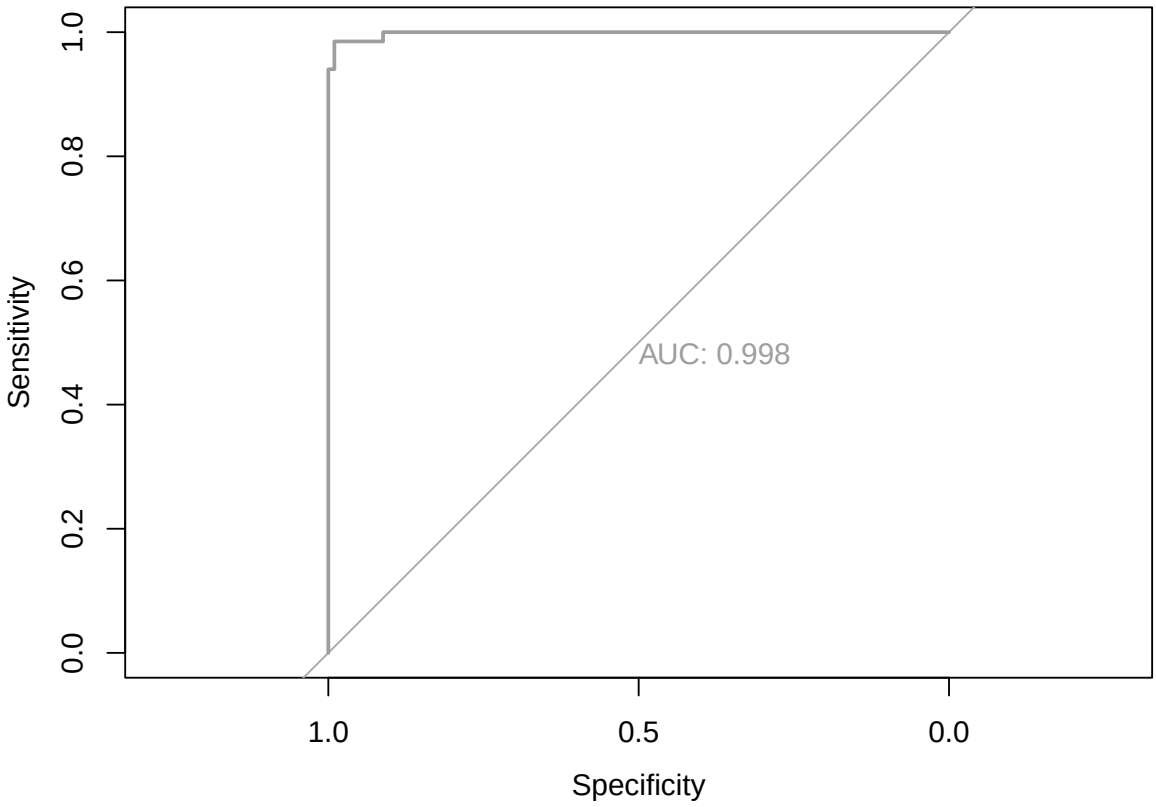

Figure 4: RoC curve for regularized linear fit

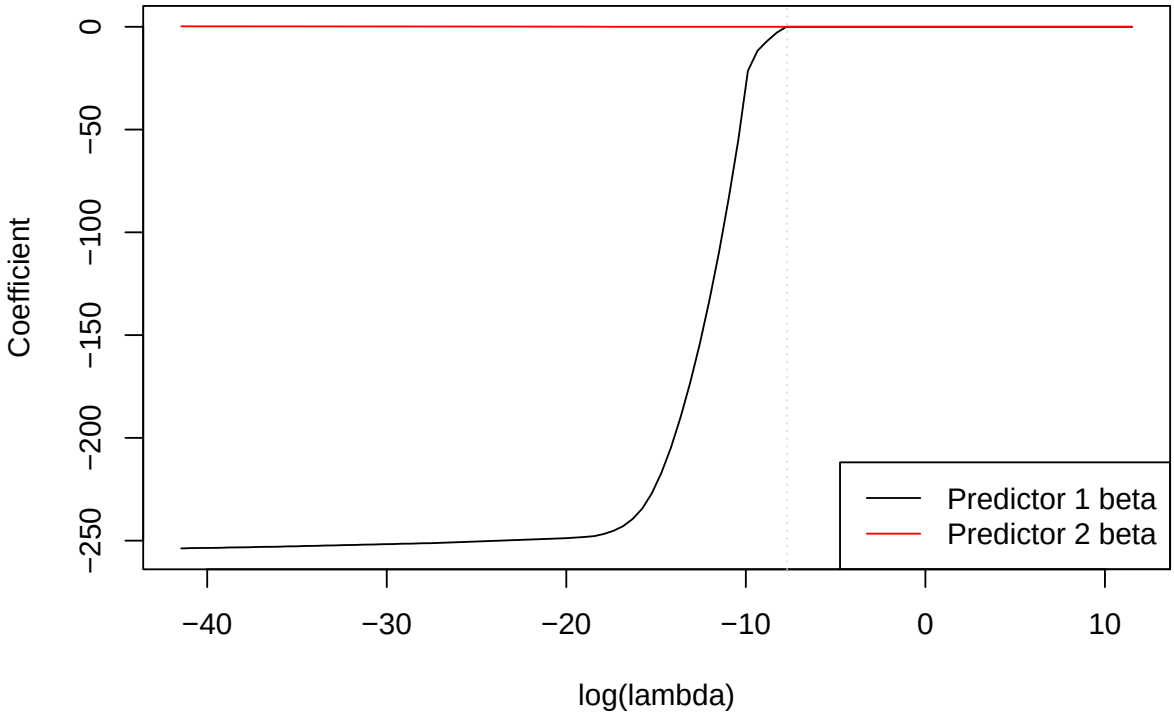

Figure 5: log(lambda) Vs beta values for predictor 1 and predictor 3 - Lasso

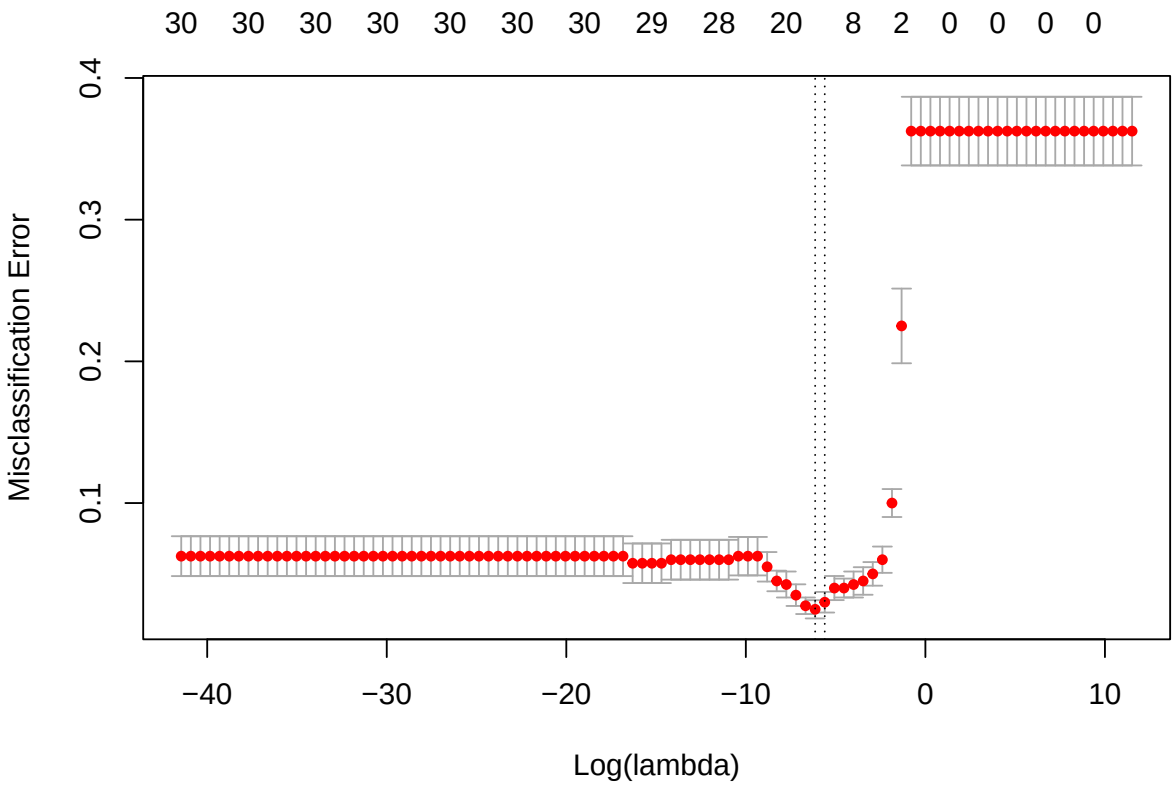

Figure 6: isclassification Error vs. Log(lambda)

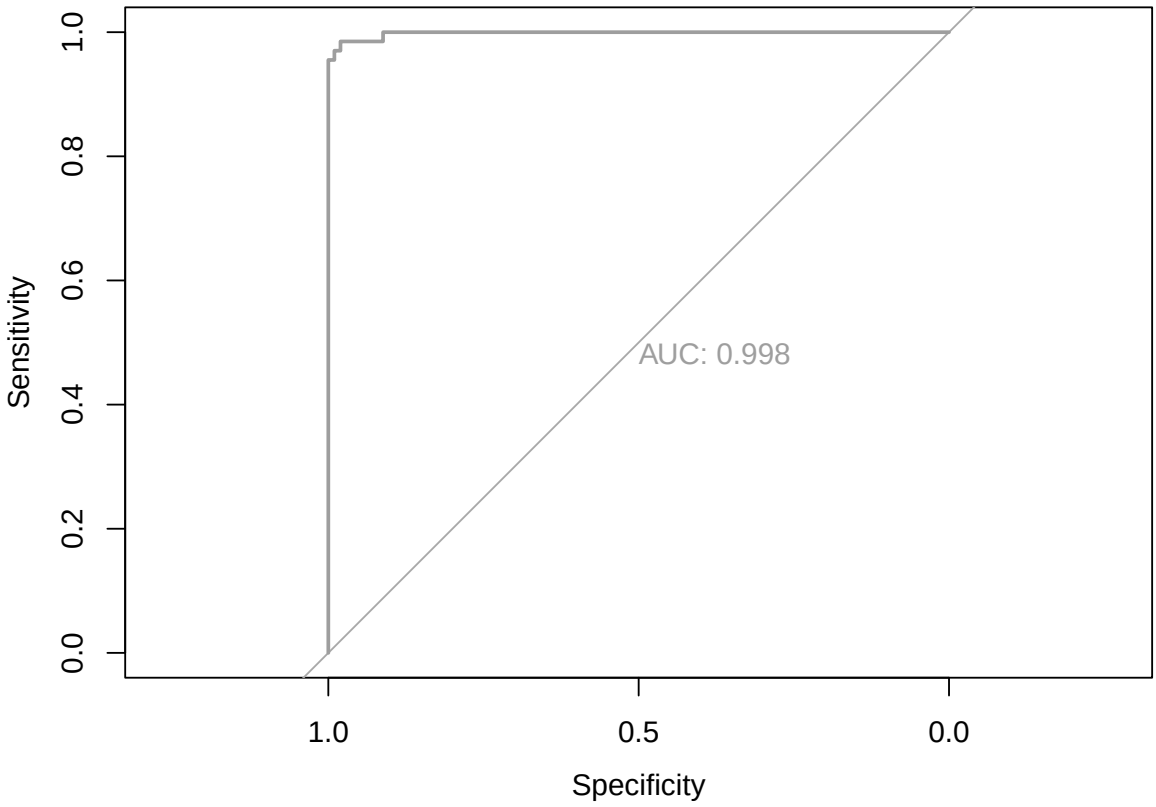

Figure 7: RoC curve for Lasso fit

Question 4

Recap of accuracy:

- For lasso fit on the test data the accuracy is 98.81%, sensitivity 100% and specificity 98.08%. Whereas, on train data, accuracy is 98.75%, sensitivity 99.30% and specificity 98.45%.
- For ridge fit on the test data the accuracy is 98.80%, sensitivity 100% and specificity 98.10%. Whereas, in the train data the accuracy is 99.0%, sensitivity 99.3% and specificity 98.8%.
- For simple logistic fit the accuracy, sensitivity, and specificity of the model on train data are estimated to be 100%. Whereas, on test data the test accuracy is 94.08%, sensitivity 89.55%, and specificity 97.06%.

First thing first, the simple logistic fit is the worst. The lasso and ridge have similar (not identical) performance, in terms of test accuracy. I guess for me given the closeness in their value of test accuracy, specificity and sensitivity, I would say one can perform as best as the other one. However, if I had to choose one, ridge tends to over fit (just a tiny amount), as compared to lasso. On related note, lasso only uses 15 (half of the predictors), where as ridge uses all the predictors (even though, the magnitude of the impact varies across each predictor). However, logistic regression would be more suitable if we are more concerned about interpretation. Unpopular thought, never ever, the use of more complicated (advanced) machine learning models be compromised by the fear of interaprebaility. In fact, when comparing conventional statistics to machine learning, the only real concern should be the quality of the data being used. If the data is of sufficient quality to allow for the application of statistical methods, then it is undoubtedly suitable for machine learning models as well.

Many researchers may believe that their research objectives and questions are better suited for statistical models, such as finding odds ratios or relative risk. However, the increasing popularity of AI methods for pattern detection and other applications is quickly rendering such generalized metrics obsolete. By utilizing these new AI tools, researchers can unlock insights that were previously unattainable, allowing for a deeper understanding of complex data sets.

Therefore, we should not be afraid to explore the full potential of advanced machine learning models. By embracing these powerful tools and adopting a data-driven approach, we can unlock new insights and discoveries that will help us tackle some of the world's most pressing challenges.

(*the accuracy values indicated in question 4 might vary from the rest of my answer as they are static values. I realized I have been using the same variable across all models, so here I had to manually add them before the PDF was generated.*)

Code Appendix

```
knitr::opts_chunk$set(fig.pos = 'H')
### Setting up the packages
library(knitr)
knitr::opts_chunk$set(echo = FALSE)
# check if packages are installed; if not, install them
packages <- c("tidyverse", "readr", "ggExtra", "plotly",
              "ggplot2","ggstatsplot","ggside","rigr","nlme","lmtest",
              "sandwich","gridExtra","broom","janitor","ellipse","caret",
              "pROC","MASS","class","purrr","tidyr","ggcorrplot","glmnet")
not_installed <- setdiff(packages, rownames(installed.packages()))
if (length(not_installed)) install.packages(not_installed)

# load packages
library(sandwich)
library(ggcorrplot)
library(glmnet)
library(janitor)
library(readr)
library(lmtest)
library(class)
library(pROC)
library(nlme)
library(ellipse)
library(broom)
library(ggstatsplot)
library(ggside)
library(caret)
library(rigr)
library(ggExtra)
library(gridExtra)
library(purrr)
library(plotly)
library(ggplot2)
library(MASS)
library(tidyverse)
library(tidyr)

### ------------------------------------------------------------
#Loading working directory of the raw data

#Please load your data/directory by changing it with your work directory
#Throughout this code module you will see a tone of places, where
#data is read and written, so please make sure to change them to your
#working directory folder format

working_directory_data <- setwd("C:/Users/latera/Desktop/ML_ass")

#loads the data on a variable df
study_data <- read.csv("data/wdbc.data", header=FALSE) %>%
  dplyr::select(diagnosis = V2, P_ = V3:V32) %>%
  mutate(diagnosis = factor(diagnosis, levels = c("B", "M"),
```

```
                            labels = c("Benign", "Malignant")))

#Describe the data
#as.tibble(study_data)


#Check if the outcome is factor
is.factor(study_data$diagnosis)

summary <- study_data %>%
  group_by(diagnosis) %>%
  dplyr::summarise(observations = n())

knitr::kable(summary, caption = "Summary table of the data")
set.seed(2)
study_data_idx = sample(nrow(study_data), 400)
study_data_trn = study_data[study_data_idx, ]
study_data_tst = study_data[-study_data_idx, ]
normalize_predictors <- function(df) {
  # Get the column names of the predictors
  predictor_cols <- names(df)[2:31]

  # Loop over the predictors and normalize each one
  for (col in predictor_cols) {
    df[[col]] <- (df[[col]] - mean(df[[col]])) / sd(df[[col]])
  }

  return(df)
}

# Apply normalization to the training set
study_data_trn <- normalize_predictors(study_data_trn)

# Apply normalization to the test set
study_data_tst <- normalize_predictors(study_data_tst)

compute_corr_matrix <- function(df) {
  # Select all columns except the first one (assumed to be the response variable)
  X_cols <- 2:ncol(df)

  # Compute the correlation matrix on the predictors
  corr_mat <- cor(df[, X_cols])

  # Plot the correlation matrix using ggcorrplot
  ggcorrplot(corr_mat, type = "lower", lab = FALSE,
             title = "Correlation Matrix of Predictors",
             ggtheme = ggplot2::theme_gray,
             legend.title = "Correlation scale")
}

compute_corr_matrix(study_data_trn)
model_glm = glm(diagnosis ~ ., data = study_data_trn,
                family = "binomial")
```

```r
model_glm_summary <- summary(model_glm, correlation = TRUE)

knitr::kable(
  coef(model_glm_summary),
  digits = 2,
  caption = "Logistic regression training coefficients"
)

#Be bayesed, this only works for type dataframe, not matrix
create_bayes_classifier <- function(model_fit) {

  function(data, thresholds) {

    add_class <- function(threshold) {

      classes_predicted <- factor(
        model_fit(data) > threshold,
        levels = c(FALSE, TRUE),
        labels = c("Benign", "Malignant")
      )

      data %>% dplyr::mutate(
        classes_predicted = classes_predicted,
        threshold = threshold
      )

    }

    thresholds %>% purrr::map_dfr(add_class)
  }

}
log_bayes_pred <- create_bayes_classifier(
  function(data) predict(model_glm, data, type = "response")
)
train_pred = log_bayes_pred(study_data_trn, 0.5)
train_tab = table(train_pred$classes_predicted, study_data_trn$diagnosis)
train_con_mat = confusionMatrix(train_tab, positive = "Malignant")

knitr::kable(train_tab,
      col.names = c("Actual Benign", "Actual Malignant"),
      digits = 5,caption = "Confussion matrix for train data")

knitr::kable(c(train_con_mat$overall["Accuracy"],
train_con_mat$byClass["Sensitivity"],
train_con_mat$byClass["Specificity"],
train_con_mat$byClass["Pos Pred Value"],
train_con_mat$byClass["Neg Pred Value"],
train_con_mat$byClass["Prevalence"],
train_con_mat$byClass["Detection Rate"]),
      col.names = c("Percentages"),
      digits = 5,caption = "Confussion matrix for
train data")
```

```
trest_pred = log_bayes_pred(study_data_tst, 0.5)
test_tab = table(trest_pred$classes_predicted, study_data_tst$diagnosis)
test_con_mat = confusionMatrix(test_tab, positive = "Malignant")

knitr::kable(test_tab,
      col.names = c("Actual Benign", "Actual Malignant"),
      digits = 5,caption = "Confussion matrix for test data")

knitr::kable(c(test_con_mat$overall["Accuracy"],
test_con_mat$byClass["Sensitivity"],
test_con_mat$byClass["Specificity"],
test_con_mat$byClass["Pos Pred Value"],
test_con_mat$byClass["Neg Pred Value"],
test_con_mat$byClass["Prevalence"],
test_con_mat$byClass["Detection Rate"]),
      col.names = c("Percentages"),
      digits = 5,caption = "Confussion matrix for
test data")
X_study_trn <- study_data_trn %>% select(-diagnosis) %>% as.matrix()
X_study_tst <- study_data_tst %>% select(-diagnosis) %>% as.matrix()

Y_study_trn <- study_data_trn$diagnosis
Y_study_tst <- study_data_tst$diagnosis
# Create a grid of values for lambda
lambda_seq <- 10^seq(5, -18, length = 100)
# Fit the ridge logistic regression model
ridge_fit <- glmnet(X_study_trn, Y_study_trn, family = "binomial", alpha = 0,
                    lambda = lambda_seq)



coef_P1 <- coef(ridge_fit)["P_1", ]
coef_P3 <- coef(ridge_fit)["P_3", ]

# Plot the coefficients in function of log(lambda)
plot(log(ridge_fit$lambda), coef_P1, type = "l", xlab = "log(lambda)",
     ylab = "Coefficient")
lines(log(ridge_fit$lambda), coef_P3, type = "l", col = "red")
legend("bottomright", legend = c("Predictor 1 beta", "Predictor 2 beta"),
       col = c("black", "red"), lty = 1)
abline(v=-10, col = "lightgray", lty = 3)
cvfit <- cv.glmnet(X_study_trn, Y_study_trn, family = "binomial",
                   type.measure = "class",
                   alpha = 0, lambda = lambda_seq)

# Find optimal lambda value that minimizes CV error
opt_lambda <- cvfit$lambda.min
#cat("Optimal lambda value:", opt_lambda, "\n")

# Plot misclassification error vs. log(lambda)
plot(cvfit, xlab = "Log(lambda)")
# Print the selected lambda value
#cat("Selected lambda value:", lambda_min, "\n")
```

```r
# Extract the coefficients of the selected model
coef_sel <- coef(cvfit, s = opt_lambda)


#print(coef_sel)

knitr::kable(
  coef_sel[-3, ],
  digits = 5,
  col.names=c("beta values"),
  caption = "Coefficients of the selected model, with CV"
)
fit_glmnet <- glmnet(x=X_study_trn, y=Y_study_trn, family = "binomial", alpha = 0,
              lambda = opt_lambda)

pred_train <- predict(fit_glmnet, newx = X_study_trn, type = "response")
Y_train_pred <- ifelse(pred_train > 0.5, "Malignant", "Benign")
train_tab = table(Y_study_trn, Y_train_pred)
train_con_mat = confusionMatrix(train_tab, positive = "Malignant")

knitr::kable(train_tab,
      col.names = c("Actual Benign", "Actual Malignant"),
      digits = 5,caption = "Confussion matrix for train data")

knitr::kable(c(train_con_mat$overall["Accuracy"],
train_con_mat$byClass["Sensitivity"],
train_con_mat$byClass["Specificity"],
train_con_mat$byClass["Pos Pred Value"],
train_con_mat$byClass["Neg Pred Value"],
train_con_mat$byClass["Prevalence"],
train_con_mat$byClass["Detection Rate"]),
      col.names = c("Percentages"),
      digits = 5,caption = "Confussion matrix for
train data")
pred_tst <- predict(fit_glmnet, newx = X_study_tst, type = "response")
Y_tst_pred <- ifelse(pred_tst > 0.5, "Malignant", "Benign")
tst_tab = table(Y_study_tst, Y_tst_pred)
tst_con_mat = confusionMatrix(tst_tab, positive = "Malignant")

knitr::kable(tst_tab,
      col.names = c("Actual Benign", "Actual Malignant"),
      digits = 5, caption = "Confussion matrix for test data")

knitr::kable(c(tst_con_mat$overall["Accuracy"],
tst_con_mat$byClass["Sensitivity"],
tst_con_mat$byClass["Specificity"],
tst_con_mat$byClass["Pos Pred Value"],
tst_con_mat$byClass["Neg Pred Value"],
tst_con_mat$byClass["Prevalence"],
tst_con_mat$byClass["Detection Rate"]),
      col.names = c("Percentages"),
      digits = 5,caption = "Confussion matrix for
test data")
```

```r
test_roc = roc(Y_study_tst ~ pred_tst, plot = TRUE,
               print.auc = TRUE, col=8)

# Create a grid of values for lambda
lambda_seq <- 10^seq(5, -18, length = 100)
# Fit the ridge logistic regression model
lasso_fit <- glmnet(X_study_trn, Y_study_trn, family = "binomial", alpha = 1,
                    lambda = lambda_seq)


coef_P1 <- coef(lasso_fit)["P_1", ]
coef_P3 <- coef(lasso_fit)["P_3", ]

# Plot the coefficients in function of log(lambda)
plot(log(lasso_fit$lambda), coef_P1, type = "l", xlab = "log(lambda)",
     ylab = "Coefficient")
lines(log(lasso_fit$lambda), coef_P3, type = "l", col = "red")
legend("bottomright", legend = c("Predictor 1 beta", "Predictor 2 beta"),
       col = c("black", "red"), lty = 1)
abline(v=-7.7, col = "lightgray", lty = 3)
cvfit <- cv.glmnet(X_study_trn, Y_study_trn, family = "binomial",
                   type.measure = "class",
                   alpha = 1, lambda = lambda_seq)

# Find optimal lambda value that minimizes CV error
opt_lambda <- cvfit$lambda.min
#cat("Optimal lambda value:", opt_lambda, "\n")

# Plot misclassification error vs. log(lambda)
plot(cvfit, xlab = "Log(lambda)")
# Print the selected lambda value
#cat("Selected lambda value:", lambda_min, "\n")

# Extract the coefficients of the selected model
coef_sel <- coef(cvfit, s = opt_lambda)


#print(coef_sel)

knitr::kable(
  coef_sel[-3, ],
  digits = 5,
  col.names=c("beta values"),
  caption = "Coefficients of the selected model, with CV"
)
fit_glmnet <- glmnet(x=X_study_trn, y=Y_study_trn, family = "binomial", alpha = 1,
              lambda = opt_lambda)

pred_train <- predict(fit_glmnet, newx = X_study_trn, type = "response")
Y_train_pred <- ifelse(pred_train > 0.5, "Malignant", "Benign")
train_tab = table(Y_study_trn, Y_train_pred)
train_con_mat = confusionMatrix(train_tab, positive = "Malignant")
```

```
knitr::kable(train_tab,
      col.names = c("Actual Benign", "Actual Malignant"),
      digits = 5,caption = "Confussion matrix for train data")

knitr::kable(c(train_con_mat$overall["Accuracy"],
train_con_mat$byClass["Sensitivity"],
train_con_mat$byClass["Specificity"],
train_con_mat$byClass["Pos Pred Value"],
train_con_mat$byClass["Neg Pred Value"],
train_con_mat$byClass["Prevalence"],
train_con_mat$byClass["Detection Rate"]),
      col.names = c("Percentages"),
      digits = 5,caption = "Confussion matrix for
train data")
pred_tst <- predict(fit_glmnet, newx = X_study_tst, type = "response")
Y_tst_pred <- ifelse(pred_tst > 0.5, "Malignant", "Benign")
tst_tab = table(Y_study_tst, Y_tst_pred)
tst_con_mat = confusionMatrix(tst_tab, positive = "Malignant")

knitr::kable(tst_tab,
      col.names = c("Actual Benign", "Actual Malignant"),
      digits = 5, caption = "Confussion matrix for train data")

knitr::kable(c(tst_con_mat$overall["Accuracy"],
tst_con_mat$byClass["Sensitivity"],
tst_con_mat$byClass["Specificity"],
tst_con_mat$byClass["Pos Pred Value"],
tst_con_mat$byClass["Neg Pred Value"],
tst_con_mat$byClass["Prevalence"],
tst_con_mat$byClass["Detection Rate"]),
      col.names = c("Percentages"),
      digits = 5,caption = "Confussion matrix for
train data")

test_roc = roc(Y_study_tst ~ pred_tst, plot = TRUE,
               print.auc = TRUE, col=8)
```