Assignment 2 Solution

Biomedical Data Science (MATH11174), 22/23, Semester 2

Reproduced by Johnny MyungWon Lee

April 6, 2023

Due on Thursday, 6th of April 2023, 5:00pm

! Pay Attention

The assignment is marked out of 100 points, and will contribute to ***30%*** of your final mark. The aim of this assignment is to produce a precise report in biomedical studies with the help of statistical and machine learning. Please complete this assignment using **Quarto/Rmarkdown file and render/knit this document only in PDF format** (rendering while solving the questions will prevent sudden panic before submission!). Submit using the **gradescope link on Learn** and ensure that **all questions are tagged accordingly**. You can simply click render on the top left of Rstudio (`Ctrl+Shift+K`). If you cannot render/knit to PDF directly, open **Terminal** in your RStudio (`Alt+Shift+R`) and type `quarto tools install tinytex`, otherwise please follow this link. If you have any code that does not run you will not be able to render nor knit the document so comment it as you might still get some grades for partial code.
Clear and reusable code will be rewarded. Codes without proper indentation, choice of variable identifiers, **comments**, efficient code, etc will be penalised. An initial code chunk is provided after each subquestion but **create as many chunks as you feel is necessary** to make a clear report. Add plain text explanations in between the chunks when required to make it easier to follow your code and reasoning. Ensure that all answers containing multiple values should be presented and formatted only with `kable()` and `kable_styling()` otherwise penalised (**no use of `print()` or `cat()`**). All plots must be displayed with clear title, label and legend otherwise penalised.

Problem 1 (27 points)

File `wdbc2.csv` (available from the accompanying zip folder on Learn) refers to a study of breast cancer where the outcome of interest is the type of the tumour (benign or malignant, recorded in column `diagnosis`). The study collected 30 imaging biomarkers on 569 patients.

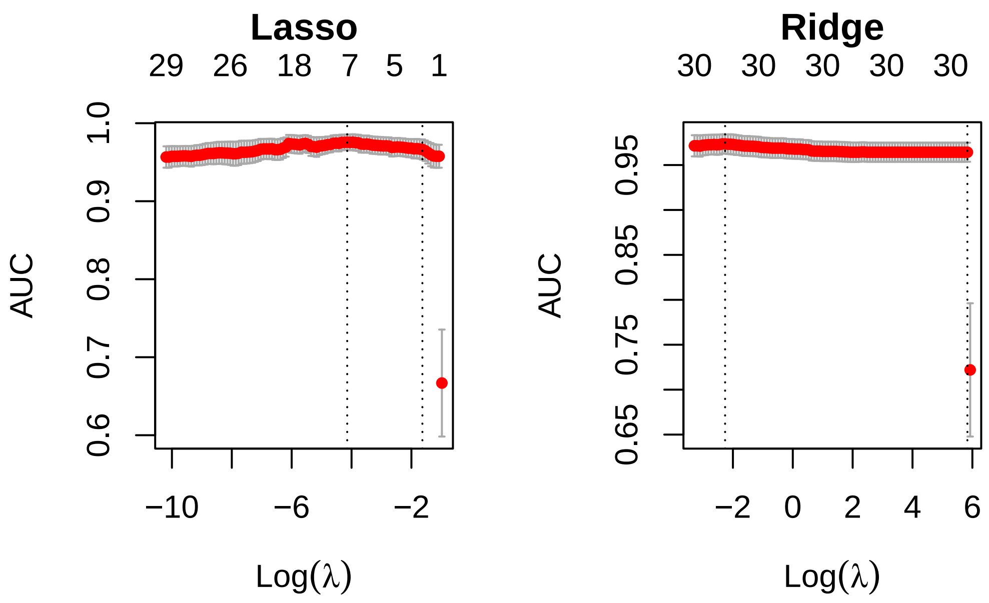
Problem 1.a (7 points)

- Using package `caret`, create a data partition so that the training set contains 70% of the observations (set the random seed to 984065 beforehand).
- Fit both a ridge and Lasso regression model which use cross validation on the training set to diagnose the type of tumour from the 30 biomarkers.
- Then use a plot to help identify the penalty parameter λ that maximises the AUC and report the λ for both ridge and Lasso regression using `kable()`.
- ***Note : there is no need to use the*** `prepare.glmnet()` ***function from lab 4, using*** `as.matrix()` ***with the required columns is sufficient.***

```r
wdbc.dt <- fread("data_assignment2/wdbc2.csv")
## Creating partition
set.seed(984065)
train.idx <-createDataPartition(wdbc.dt$diagnosis, p = 0.7)$Resample1
## Edit the outcome to be a factor (binary), drop patient id
wdbc.dt <- wdbc.dt[, !"id", with = FALSE]
wdbc.dt$diagnosis <- factor(wdbc.dt$diagnosis,
                      levels=c("benign","malignant"))
## Split into train and test subsets
wdbc.train.dt <- wdbc.dt[train.idx,]
wdbc.test.dt <- wdbc.dt[!train.idx,]
## Store the outcome and the covariates separately
y.wdbc.train.dt <- wdbc.train.dt$diagnosis
x.wdbc.train.dt <- as.matrix(wdbc.train.dt[, !"diagnosis", with = FALSE])
y.wdbc.test.dt <- wdbc.test.dt$diagnosis
x.wdbc.test.dt <- as.matrix(wdbc.test.dt[, !"diagnosis", with = FALSE])
```

```r
## Fit lasso and ridge models
fit.cv.lasso <- cv.glmnet(x.wdbc.train.dt, y.wdbc.train.dt,
                family = "binomial", type.measure = "auc", alpha = 1)
fit.cv.ridge <- cv.glmnet(x.wdbc.train.dt, y.wdbc.train.dt,
                family = "binomial", type.measure = "auc", alpha = 0)
```

```r
## Plot the cross-validation curves
par(mfrow = c(1,2), mar = c(4,4,5,2))
plot(fit.cv.lasso, main = "Lasso")
plot(fit.cv.ridge, main = "Ridge")
```



```r
## Print true lambda values
lasso.lambda <- fit.cv.lasso$lambda.min
ridge.lambda <- fit.cv.ridge$lambda.min
kable(data.table(lasso.lambda, ridge.lambda),
      caption = "Penalty Parameter that maximises AUC") |>
  kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 1: Penalty Parameter that maximises AUC

lasso.lambda	ridge.lambda
0.0158513	0.1042908

The left-most dotted line in each cross-validation curve indicates the λ parameter that maximises AUC. The two values are approximately $\exp(-4.1)$ for the lasso model and $\exp(-2.3)$ for the ridge model respectively. The exact values can be seen at Table 1.

Problem 1.b (2 points)

- Create a data table that for each value of `lambda.min` and `lambda.1se` for each model fitted in **problem 1.a** that contains the corresponding λ, AUC and model size.
- Use 3 significant figures for floating point values and comment on these results.
- ***Note: The AUC values are stored in the field called*** `cvm`.

```r
## Find indices of lambdas that maximizes AUC and w/in 1 SE of max AUC
idx.lasso.min <- fit.cv.lasso$index["min",]
idx.lasso.1se <- fit.cv.lasso$index["1se",]
idx.ridge.min <- fit.cv.ridge$index["min",]
idx.ridge.1se <- fit.cv.ridge$index["1se",]

lambda.vals <- c(signif(fit.cv.lasso$lambda.min, 3),
                      signif(fit.cv.ridge$lambda.min, 3),
                      signif(fit.cv.lasso$lambda.1se, 3),
                      signif(fit.cv.ridge$lambda.1se, 3))
auc.vals <- signif(c(fit.cv.lasso$cvm[idx.lasso.min],
                    fit.cv.ridge$cvm[idx.ridge.min],
                    fit.cv.lasso$cvm[idx.lasso.1se],
                    fit.cv.ridge$cvm[idx.ridge.1se]), 3)
model.size <- signif(c(fit.cv.lasso$nzero[idx.lasso.min],
                      fit.cv.ridge$nzero[idx.ridge.min],
                      fit.cv.lasso$nzero[idx.lasso.1se],
                      fit.cv.ridge$nzero[idx.ridge.1se]), 3)

dt <- data.table(model = c("Lasso.min","Ridge.min", "Lasso.1se", "Ridge.1se"),
                 lambda = lambda.vals,
                 AUC = auc.vals,
                 Model.Size = model.size)

kable(dt, caption = "Lambda values with its model size and AUC") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 2: Lambda values with its model size and AUC

model	lambda	AUC	Model.Size
Lasso.min	0.0159	0.976	7
Ridge.min	0.1040	0.973	30
Lasso.1se	0.1950	0.966	2
Ridge.1se	342.0000	0.964	30

λ is a penalty parameter that shrinks the coefficient to zero for Lasso and near zero for Ridge.

This is the reason why, Lasso has a smaller model size compared to Ridge regression. From Table 2, Lasso with $\lambda = 0.0159$ will have model size of 7 and Ridge with $\lambda = 0.104$ will have model size of 30 (unchanged). Looking at Table 1. we can see that `lambda.min` for both Lasso and Ridge regression have low AUC values compared to `lambda.1se`. This is because, the `lambda.min` values maximises the AUC values and minimises the error. Then, we compare the model accuracy among Lasso and Ridge. Lasso has the AUC value of 0.976 and Ridge has 0.973. From this, Lasso regression represents the training dataset better than Ridge.

Problem 1.c (7 points)

- Perform both backward (we denote this as **model B**) and forward (**model S**) stepwise selection on the same training set derived in **problem 1.a**. Mute all the trace by setting `trace = FALSE`.
- Report the variables selected and their standardised regression coefficients in increasing order of the absolute value of their standardised regression coefficient.
- Discuss the results and how the different variables entering or leaving the model influenced the final result.
- ***Note : you can mute the warning by assigning `{r warning = FALSE}`***

```r
## Standardize all variables
wdbc.train.scaled.dt <- wdbc.train.dt %>% copy()
covar.colnames <- colnames(wdbc.train.dt[,-c("diagnosis")])
wdbc.train.scaled.dt <- wdbc.train.scaled.dt[,
              (covar.colnames) := lapply(.SD, function(x) x / sd(x)),
              .SDcols = covar.colnames]
## Training set standard deviations (store to use later for test set)
wdbc.train.df <- as.data.frame(copy(wdbc.train.dt))
wdbc.train.sd <- apply(wdbc.train.df[, (covar.colnames)], 2, sd)
```

```r
## Define full model and null model
full.model <- glm(diagnosis~., data = wdbc.train.scaled.dt, family = "binomial")
null.model <- glm(diagnosis~1, data = wdbc.train.scaled.dt, family = "binomial")
## Perform forward and backward stepwise selection
model.B <- stepAIC(full.model, direction = "back", trace = FALSE)
model.S <- stepAIC(null.model, scope = list(upper = full.model),
direction = "forward", trace = FALSE)
```

```r
## Setting coefficients in decreasing order
B.order <- order(abs(model.B$coefficients), decreasing = FALSE)
S.order <- order(abs(model.S$coefficients), decreasing = FALSE)
```

```r
## Tabulating coefficients for each model
table <- list(model.B$coefficients[B.order],
              model.S$coefficients[S.order])

kable(table, col.names = "Coefficient",
      caption = "Coefficients of Model B and Model S") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 3: Coefficients of Model B and Model S

	Coefficient		Coefficient
texture.stderr	-1.557596	smoothness	0.7783217
perimeter	2.269595	symmetry	-1.1376722
area	-2.596555	symmetry.stderr	-1.3323836
texture.worst	2.834272	perimeter.stderr	-1.3580723
fractaldimension.worst	2.884683	texture	1.6545301
fractaldimension.stderr	-3.109179	area	-1.6689807
compactness.stderr	-3.226191	concavity.worst	1.7848530
radius.stderr	4.877092	compactness.worst	-2.3102578
concavity	-4.963479	symmetry.worst	2.3111535
concavepoints	5.500309	concavepoints.worst	2.3500713
concavity.stderr	8.280173	radius.stderr	4.7830443
radius.worst	17.144240	area.worst	-15.8754542
area.worst	-17.281793	radius.worst	17.6173905
(Intercept)	-66.041721	(Intercept)	-54.4588699

The process of backward elimination is performed, starting with a full model that includes all covariates. From there, the removal of each covariate is tested against the others, by comparing the AIC values for a model without that covariate (including no removal, using the current model). The resulting model with the lowest AIC is selected, and that covariate removed. The process then begins again with the updated model. This process is repeated until the option of not removing any covariate produces a model with the lowest AIC score. On the first iteration, it is seen that removing the covariate `smoothness` will produce a model with the lowest AIC score. This is removed and then on the next iteration the covariate `compactness.worst` is then selected for removal. The process is repeated and the covariates that are removed can be seen in the output below; the top covariate is selected for removal on each iteration. It is interesting to note that **Model B** has excluded `concavepoints.worst` from the model on the fourth iteration, considering that Lasso regression found this covariate to be highly important in determining tumor type.

It is interesting to see that it contains only three of the seven biomarkers that were selected by the Lasso regression model (for AUC within one standard error of maximum). The biomarkers of `radius.worst` and `area.worst` appear to have the largest impact in model B. Forward selection is now performed, where covariates are individually added to a null model, the AIC score evaluated for each model, and the model with the lowest AIC score selected. The process is then repeated with the updated model, and the top covariate in each iteration below is selected for addition to the model.

Overall, the **Model B** and **Model S** each selected 13 biomarkers, but only four biomarkers in common: `area.worst`, `radius.worst`, `radius.stderr`, and `area`. Both models share the top two most impactful covariates, but the process of the different variables leaving the model in backwards elimination and entering the **Model S** has resulted in different coefficient values and nine non-shared covariates being selected for the model. This demonstrates that the selection process used can have a significant impact on final model choice when there are many covariates.

Problem 1.d (3 points)

- Compare the goodness of fit of **model B** and **model S**
- Interpret and explain the results you obtained.
- Report the values using `kable()`.

```r
## Computing the AIC and BIC values of each model
dt <- data.table(c("Model B AIC", "Model S AIC", "Model B BIC", "Model S BIC"),
                 c(AIC(model.B), AIC(model.S), BIC(model.B), BIC(model.S)))
colnames(dt) <- c("Goodness of fit", "Values")
kable(t(dt), caption = "Goodness of fit for Model B and Model S") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 4: Goodness of fit for Model B and Model S

Goodness of fit	Model B AIC	Model S AIC	Model B BIC	Model S BIC
Values	110.3510	120.2284	166.1964	176.0739

```r
## Testing goodness of fit of model B and model S
chisq.b <- pchisq(model.B$null.deviance - model.B$deviance,
                  df = 12, lower.tail = FALSE)
chisq.s <- pchisq(model.S$null.deviance - model.S$deviance,
                  df = 14, lower.tail = FALSE)

dt <- data.table(c("Model B", "Model S"),
                  c(chisq.b, chisq.s))
colnames(dt) <-c("", "p-value")
```

```r
kable(t(dt), caption = "Goodness of fit for Model B and Model S") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 5: Goodness of fit for Model B and Model S

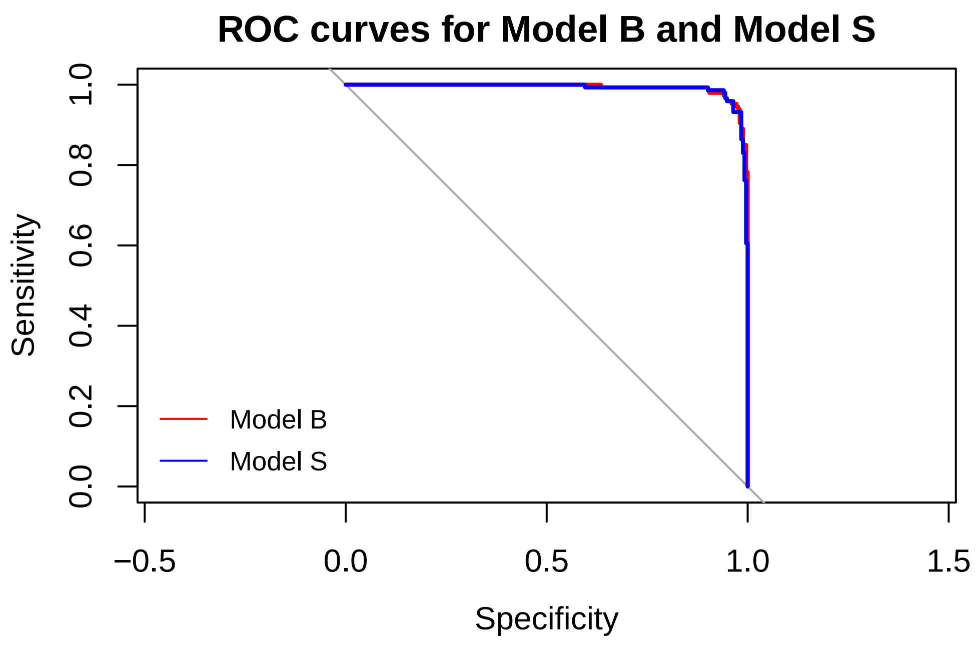
	Model B	Model S
p-value	3.157708e-87	1.428145e-83

Model B has a AIC value of 110.3509893 and **Model S** has a AIC value of 120.2284352. **Model B** has a lower AIC value thus a better model. To further validate this, we can also refer to the result from χ^2-test. We obtained the p-value for each model where Model B has 0 and **Model S** has 0. Since **Model B** has a lower p-value, we can conclude that **Model B** is a better fit to the training dataset.

Problem 1.e (2 points)

- Plot the ROC curve of the trained model for both **model B** and **model S**. Display with clear title, label and legend.
- Report AUC values in 3 significant figures for both **model B** and **model S** using `kable()`.
- Discuss which model has a better performance.

```r
## Data frame for actual observations for the outcome in train set,
## and predicted outcome from the model
pred.model.B <- data.frame(obs = wdbc.train.scaled.dt$diagnosis,
pred = predict(model.B, newdata = wdbc.train.scaled.dt, type = "response"))
## Repeat process for model S
pred.model.S <- data.frame(obs = wdbc.train.scaled.dt$diagnosis,
pred = predict(model.S, newdata = wdbc.train.scaled.dt, type = "response"))

## Calculate the predictive ability of the training models
suppressMessages(invisible({
  roc.B <- roc(obs ~ pred, data = pred.model.B, plot = TRUE, xlim = c(0,1),
              col = "red", main = "ROC curves for Model B and Model S")
  roc.S <- roc(obs ~ pred, data = pred.model.S, plot = TRUE, xlim = c(0,1),
              col = "blue", add = TRUE)
  legend("bottomleft", legend = c("Model B", "Model S"),
            col = c("red", "blue"), lty = 1, cex = 0.8, bty = "n")
}))
```



```r
## Tabulating the AUC values for Model B and Model S
dt <- data.table(c("Model B", "Model S"),
                 c(signif(roc.B$auc, 3), signif(roc.S$auc, 3)))
colnames(dt) <-c("", "AUC")
kable(t(dt), caption = "AUC for Model B and Model S") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 6: AUC for Model B and Model S

	Model B	Model S
AUC	0.993	0.991

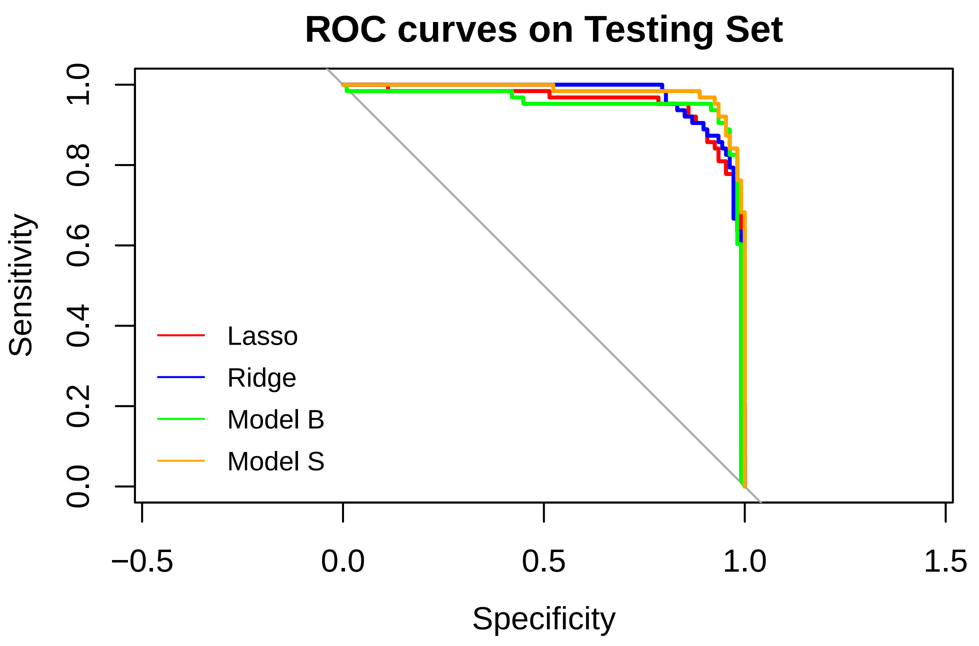
We computed the ROC curve using `roc()`, to compare the two models, **Model B** and **Model S**. Visually, we could not tell much difference. However, looking at Table 6, we can see that both models have high accuracy with the values of 0.993 and 0.991 respectively. Since **Model B** has a slight higher value, like **problem 1.a**, we can conclude that **Model B** is a better model.

Problem 1.f (6 points)

- Use the four models to predict the outcome for the observations in the test set (use the λ at 1 standard error for the penalised models).

- Plot the ROC curves of these models (on the sameplot, using different colours) and report their test AUCs.
- Compare the training AUCs obtained in **problems 1.b and 1.e** with the test AUCs and discuss the fit of the different models.
- Display with clear title, label and legend.

```r
## Prediction of Lasso and Ridge Regression Model
lasso.pred <- predict(fit.cv.lasso, newx = x.wdbc.test.dt,
                        s = "lambda.1se", type = "response")
ridge.pred <- predict(fit.cv.ridge, newx = x.wdbc.test.dt,
                        s = "lambda.1se", type = "response")

## Prediction of Model B and Model S
wdbc.test.dt. <- wdbc.test.dt %>% copy()
for (i in covar.colnames){
  wdbc.test.scaled.dt <- wdbc.test.dt.[,
  (i) := lapply(.SD, function(x) x / sd(x)), .SDcols = i]
}

pred.B <- predict(model.B, newdata = wdbc.test.scaled.dt,
                     type = "response")
pred.S <- predict(model.S, newdata = wdbc.test.scaled.dt,
                     type = "response")

## Plotting ROC
suppressMessages(invisible({
auc.lasso <- roc(y.wdbc.test.dt, lasso.pred, plot = TRUE,
              xlim = c(0,1), col = "red", main = "ROC curves on Testing Set")$auc
auc.ridge <- roc(y.wdbc.test.dt, ridge.pred, plot = TRUE,
              col = "blue", add = TRUE)$auc
auc.B <- roc(y.wdbc.test.dt, pred.B, plot = TRUE,
              col = "green", add = TRUE)$auc
auc.S <- roc(y.wdbc.test.dt, pred.S, plot = TRUE,
              col = "orange", add = TRUE)$auc
legend("bottomleft", legend = c("Lasso", "Ridge", "Model B", "Model S"),
col = c("red", "blue", "green", "orange"), lty = 1, cex = 0.8, bty = "n")
}))
```



```
dt <- data.table(c("Lasso", "Ridge", "Model B", "Model S"),
                 signif(c(fit.cv.lasso$cvm[idx.lasso.1se],
                   fit.cv.ridge$cvm[idx.ridge.1se],
                   roc.B$auc, roc.S$auc),3),
                 signif(c(auc.lasso, auc.ridge, auc.B, auc.S), 3))
colnames(dt) <- c("", "training AUC", "testing AUC")
kable(t(dt), caption = "Training and Testing Acuraccy of each Model") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 7: Training and Testing Acuraccy of each Model

	Lasso	Ridge	Model B	Model S
training AUC	0.966	0.964	0.993	0.991
testing AUC	0.952	0.968	0.950	0.982

Comparison of the AUC values of all four models and their prediction performance on the test set of data, reveals that the forward selection **Model S** appears to have the best performance. The backwards elimination **Model B** performed overall the best on the training data, but performed the worst of all models on the test set. This indicates there may have been some overfitting that occurred during model development. Both Lasso and ridge regression performed relatively well, with similar training and test AUC values. Ridge regression performed better on the test set, but this is unusual and likely due to chance (randomness of the train and test split). All of these AUC values are excellent, but it is seen that the forward selection **Model S** appears to have the best predictive ability.

Problem 2 (40 points)

File GDM.raw.txt (available from the accompanying zip folder on Learn) contains 176 SNPs to be studied for association with incidence of gestational diabetes (a form of diabetes that is specific to pregnant women). SNP names are given in the form rs1234_X where rs1234 is the official identifier (rsID), and X (one of A, C, G, T) is the reference allele.

Problem 2.a (3 points)

- Read in file GDM.raw.txt into a data table named gdm.dt.
- Impute missing values in gdm.dt according to SNP-wise median allele count and display first 10 rows and first 7 columns using kable().

```r
## Function to impute to median
impute.to.median <- function(x) {
  # Only apply function to numeric or integer columns
  if (is.numeric(x) || is.integer(x)){
  # Find index of missing values
  na.idx <- is.na(x)
  # Replace missing values with median of the observed values
  x[na.idx] <- median(x, na.rm=TRUE)
  }
  return(x)
}
gdm.dt <- fread("data_assignment2/GDM.raw.txt")
## Identify numeric columns, not including outcome
numcols.gdm <- gdm.dt[, .SD, .SDcols = sapply(gdm.dt, is.numeric)] %>% colnames
numcols.gdm <- numcols.gdm[!numcols.gdm %in% "pheno"]
## Impute missing values with median
gdm.dt.imputed <- gdm.dt %>% copy() %>%
                  .[, (numcols.gdm) := lapply(.SD, impute.to.median),
                  .SDcols = numcols.gdm]
kable(gdm.dt.imputed[c(1:10), c(1:7)],
      caption = "Gestational diabetes dataset") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

```r
## Drop patient id
gdm.dt.imputed <- gdm.dt.imputed[, !"ID", with = FALSE]
## Store the outcome and the covariates separately
y.gdm.dt.imputed <- gdm.dt.imputed[[2]]
```

Table 8: Gestational diabetes dataset

ID	sex	pheno	rs7513574_T	rs1627238_A	rs1171278_C	rs1137100_A
1	FALSE	0	1	0	0	2
2	FALSE	0	0	0	0	1
4	FALSE	1	2	1	1	1
5	FALSE	1	0	1	1	1
6	FALSE	1	0	1	1	1
7	FALSE	0	1	1	1	0
8	FALSE	0	0	0	0	1
12	FALSE	1	1	1	1	1
13	FALSE	1	2	0	0	2
18	FALSE	0	1	0	0	0

```r
x.gdm.dt.imputed <- gdm.dt.imputed[, 3:(ncol(gdm.dt)-1)]
```

Problem 2.b (8 points)

- Write function `univ.glm.test()` where it takes 3 arguements, `x`, `y` and `order`.
- `x` is a data table of `SNPs`, `y` is a binary outcome vector, and `order` is a boolean which takes `false` as a default value.
- The function should fit a logistic regression model for each `SNP` in `x`, and return a data table containing `SNP` names, regression coefficients, odds ratios, standard errors and p-values.
- If order is set to `TRUE`, the output data table should be ordered by increasing p-value.

```r
univ.glm.test <- function(x, y, order = FALSE){
  stopifnot(nrow(x) == length(y))
  ## Initialise data table
  output <- data.table("SNP" = character(), "intercept" = numeric(),
  "coefficients" = numeric(), "odds.ratios" = numeric(),
  "std.error" = numeric(), "p.value" = numeric())
  ## Run logistric regression on each SNP
  for (i in 1:ncol(x)){
    regr <- glm(y ~ x[[i]], family = binomial(link = "logit"))
    summarised <- coef(summary(regr))
    output <- rbind(output, list(names(x)[i], summarised[1,1],
                    summarised[2,1], exp(summarised[2,1]),
                    summarised[2,2], summarised[2,4]))
  }
  ## Case when order set as TRUE
  if(order == TRUE){
    output <- output[order(p.value)]
  }
  return(output)
}
```

This function takes in three arguments, `x` which should be a data table of `SNP`s and `y` a binary outcome vector. An optional argument `order` specifies whether results are ordered by increasing value. The function first creates an empty matrix to store results. It then collects all `SNP` names (column names), and runs a loop to build a model for each `SNP`. For each `SNP`, the column of data is regressed on the outcome variable using a logistical regression model with a logit link function. Model results are then individual stored in separate variables to make the code tidier. After this, all results for the `SNP` model are added as an additional row to the matrix of results. After the loop has finished, the matrix of results is then converted into a data.table. All columns apart from `SNP` name are then converted into a numeric to ensure that these are not accidentally interpreted as strings. If `order = TRUE`, then all model results are rearranged by increasing p-value with the smallest p-value at the top of the results.

Problem 2.c (5 points)

- Using function `univ.glm.test()`, run an association study for all the `SNP`s in `gdm.dt` against having gestational diabetes (column `pheno`) and name the output data table as `gdm.as.dt`.
- Print the first 10 values of the output from `univ.glm.test()` using `kable()`.
- For the `SNP` that is most strongly associated to increased risk of gestational diabetes and the one with most significant protective effect, report the summary statistics using `kable()` from the GWAS.
- Report the 95% and 99% confidence intervals on the odds ratio using `kable()`.

```r
## Run glm on each SNP in the gdm.dt data set
gdm.as.dt <- univ.glm.test(x = x.gdm.dt.imputed,
                           y = y.gdm.dt.imputed, order = TRUE)
kable(head(gdm.as.dt, 10),
      caption = "Logistic Regression Statistics for each SNP") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

```r
## Extracting neccesary columns for table
cols <- names(gdm.as.dt)[2:5]

```

Table 9: Logistic Regression Statistics for each SNP

SNP	intercept	coefficients	odds.ratios	std.error	p.value
rs12243326_A	-0.0445574	0.6454198	1.9067873	0.1583787	0.0000460
rs2237897_T	0.3743977	-0.4394456	0.6443936	0.1126133	0.0000953
rs2237892_C	0.3638020	-0.4042888	0.6674513	0.1108494	0.0002651
rs4506565_T	-0.0461714	0.4865711	1.6267287	0.1346281	0.0003013
rs7903146_C	-0.0297890	0.4790441	1.6145304	0.1382068	0.0005280
rs3786897_A	-0.0492264	0.4628940	1.5886649	0.1342880	0.0005668
rs7901695_T	-0.0392113	0.4877117	1.6285853	0.1422027	0.0006043
rs2287019_A	0.0081874	0.6038954	1.8292305	0.1856279	0.0011409
rs2943641_A	0.0238935	0.3734847	1.4527883	0.1533076	0.0148433
rs2383208_A	0.1978657	-0.4273597	0.6522289	0.1774728	0.0160389

```r
## Diving the data table to positive and negative coefficient
pos.coeff <- gdm.as.dt[coefficients>0]
neg.coeff <- gdm.as.dt[coefficients<0]

## Identify SNP with highest association to increased risk
max.SNP <- pos.coeff[which(pos.coeff$p.value==min(pos.coeff$p.value)),]
max.SNP <- max.SNP[, (cols) := .SD, .SDcols = cols]

## identify SNP with strongest protective effect
min.SNP <- neg.coeff[which(neg.coeff$p.value==min(neg.coeff$p.value)),]
min.SNP <- min.SNP[, (cols) := .SD, .SDcols = cols]

## Tabulating the SNP associated with increased risk and protective effect
kable(max.SNP, caption = "SNP associated with increased risk") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 10: SNP associated with increased risk

SNP	intercept	coefficients	odds.ratios	std.error	p.value
rs12243326_A	-0.0445574	0.6454198	1.906787	0.1583787	4.6e-05

```r
kable(min.SNP, caption = "SNP associated with protective effect") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

```r
## Calculate quantile value to multiply by
q95 <- qnorm(0.975) ## 1.96 for 95% confint
```

Table 11: SNP associated with protective effect

SNP	intercept	coefficients	odds.ratios	std.error	p.value
rs2237897_T	0.3743977	-0.4394456	0.6443936	0.1126133	9.53e-05

```r
q99 <- qnorm(0.995) ## 2.56 for 99% confint
## Computing confidence intervals for the most increased risk
max.coef <- max.SNP$coefficients
max.se <- max.SNP$std.error
max.confint.95 <- round(exp(max.coef + q95 * max.se*c(-1,1)), 3)
max.confint.99 <- round(exp(max.coef + q99 * max.se*c(-1,1)), 3)
info.dt <- data.table(c("SNP", "95% lower", "95% upper", "99% lower", "99% upper"),
                c(max.SNP$SNP, max.confint.95[1], max.confint.95[2],
                  max.confint.99[1], max.confint.99[2]))
## Computing Confidence Intervals for most protective effect
min.coef <- min.SNP$coefficients
min.se <- min.SNP$std.error
min.confint.95 <- round(exp(min.coef + q95 * min.se*c(-1,1)), 3)
min.confint.99 <- round(exp(min.coef + q99 * min.se*c(-1,1)), 3)
info.dt <- cbind(info.dt,
                 data.table(c(min.SNP$SNP, min.confint.95[1], min.confint.95[2],
                              min.confint.99[1], min.confint.99[2])))
colnames(info.dt) <- c("", "Increased Risk", "Protective Effect")
kable(t(info.dt), caption = "Confidence Intervals for SNP Associated") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 12: Confidence Intervals for SNP Associated

	SNP	95% lower	95% upper	99% lower	99% upper
Increased Risk	rs12243326_A	1.398	2.601	1.268	2.867
Protective Effect	rs2237897_T	0.517	0.804	0.482	0.861

The association study considering all the `SNP`s in this data set provides insight on which `SNP` is most strongly associated with an increased risk of gestational diabetes and which has the strongest protective effect. The two are then identified and differentiated by looking at the p-value and magnitude of their regression coefficients. The `SNP` with most positive coefficient and p-value represents the `SNP` associated with an increase risk of gestational diabets. The SNP with most negative coefficient and lowest p-value represents the `SNP` associated with a strongest protective effect.

Thus, the `SNP` `rs12243326_A` is most strongly associated with an increased risk of gestational diabetes, where the presence of this gene (compared to the baseline) is associated with an odds ratio of 1.907 that gestational diabetes will occur. For `rs12243326_A`, a 99% confidence interval of 1.268 and 2.867 suggests that there is a range of 33.5% and 150.3% increase in the odds-ratio of having gestational diabetes to not. This suggests that even at the lower end of the CI there is a significant effect of having the `SNP rs12243326_A` on an increased risk of gestational diabetes.

The `SNP rs2237897_T` provides the strongest protective effect against gestational diabetes, where the presence of this gene (compared to the baseline) is associated with an odds ratio of 0.644 that gestational diabetes will occur. For `rs2237897_T`, a 99% confidence interval of 0.482 to 0.861 suggests there is a range of a 25.2% to 133.7% decrease in the odds ratio of having gestational diabetes to not. This suggests that even at the upper end of the CI there is a significant effect of having the SNP `rs2237897_T` on a decreased risk of gestational diabetes.

Problem 2.d (4 points)

- Merge your GWAS results with the table of gene names provided in file `GDM.annot.txt` (available from the accompanying zip folder on Learn).
- For `SNP`s that have p-value $< 10^{-4}$ (`hit SNP`s) report `SNP` name, effect `allele`, chromosome number, corresponding `gene` name and `pos`.
- Using `kable()`, report for each `snp.hit` the names of the genes that are within a 1Mb window from the `SNP` position on the chromosome.
- ***Note: That are genes that fall within +/- 1,000,000 positions using the pos column in the dataset.***

```r
## Importing table of gene names
gdm.annot.dt <- fread("data_assignment2/GDM.annot.txt")
## Splitting SNP name and effect allele for SNP model results
new.gdm.dt <- gdm.as.dt %>% copy()
new.gdm.dt[, c("snp", "allele") := tstrsplit(new.gdm.dt$SNP, "_", fixed = TRUE)]
## Subsetting SNP_model_results table to only include SNPs wirh p-value < 1e-0.4
snp.hit <- merge(gdm.annot.dt, new.gdm.dt, by.x = "snp", by.y = "snp")
## Merging model results table with gene names table
snp.w.genes <- snp.hit[p.value < 1e-4,]
# snp.w.genes <- merge(gdm.annot.dt, snp.hit, by.x = "snp", by.y = "SNP")
# snp.w.genes <- gdm.annot.dt[snp.hit, on = .(snp = SNP)]
## Reporting SNP, effect allele, chromosome number and gene name
snp.report <- snp.w.genes[,c("snp","allele","chrom","gene", "pos")]
kable(snp.report, caption = "SNPs with a p-value < 0.0001") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 13: SNPs with a p-value < 0.0001

snp	allele	chrom	gene	pos
rs12243326	A	10	TCF7L2	114788815
rs2237897	T	11	KCNQ1	2858546

```r
# Hit SNP rs12243326
# Looking for genes that are within 1,000,000 of this SNP
gene.loc <- snp.w.genes[snp == "rs12243326", pos] # location of snp
gene.chrom <- snp.w.genes[snp == "rs12243326", chrom] # Chromosome of SNP
# Finding indices of genes that are within 1,000,000 of this position
# gene.idx <- which(abs(gene.loc - gdm.annot.dt$pos) <= 1e6)
gene.idx <- which(abs(gene.loc - snp.hit$pos) <= 1e6 & gene.chrom == snp.hit$chrom)
# name(s) of the genes that are within a 1Mb window from the SNP rs12243326
gene.within1Mb <- unique(snp.hit[gene.idx,gene])
gene.1MB.window <- matrix(gene.within1Mb, dimnames = list(c(), c("Gene Name")))
kable(gene.1MB.window, caption = "Genes within a 1Mb window of SNP rs12243326") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 14: Genes within a 1Mb window of SNP rs12243326

Gene Name
TCF7L2

```r
# Hit SNP rs2237897
# Looking for genes that are within 1,000,000 of this
gene.loc <- snp.w.genes[snp == "rs2237897", pos] # location of snp
gene.chrom <- snp.w.genes[snp == "rs2237897", chrom] # Chromosome of SNP
# Finding indices of genes that are within 1,000,000 of this position
gene.idx <- which(abs(gene.loc - snp.hit$pos) <= 1e6 & gene.chrom == snp.hit$chrom)
# names of the genes that are within a 1Mb window from the SNP rs12243326
gene.within1Mb <- unique(snp.hit[gene.idx, gene])
gene.1MB.window <- matrix(gene.within1Mb, dimnames = list(c(), c("Gene Name")))
kable(gene.1MB.window, caption = "Genes within a 1Mb window of SNP rs2237897") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 15: Genes within a 1Mb window of SNP rs2237897

Gene Name
TH
KCNQ1

As can be expected, the genes that are directly related (same row as per `gdm.annot`) to that `SNP` are displayed. For `rs12243326`, it is seen that no other genes are within a 1Mb range of that `SNP`. For `rs2237897` however, in addition to the `KCNQ1` gene that is directly related, there are also three other genes that are within a 1Mb range of that `SNP`: `CACNA2D4`, `SMG6`, and `TH`.

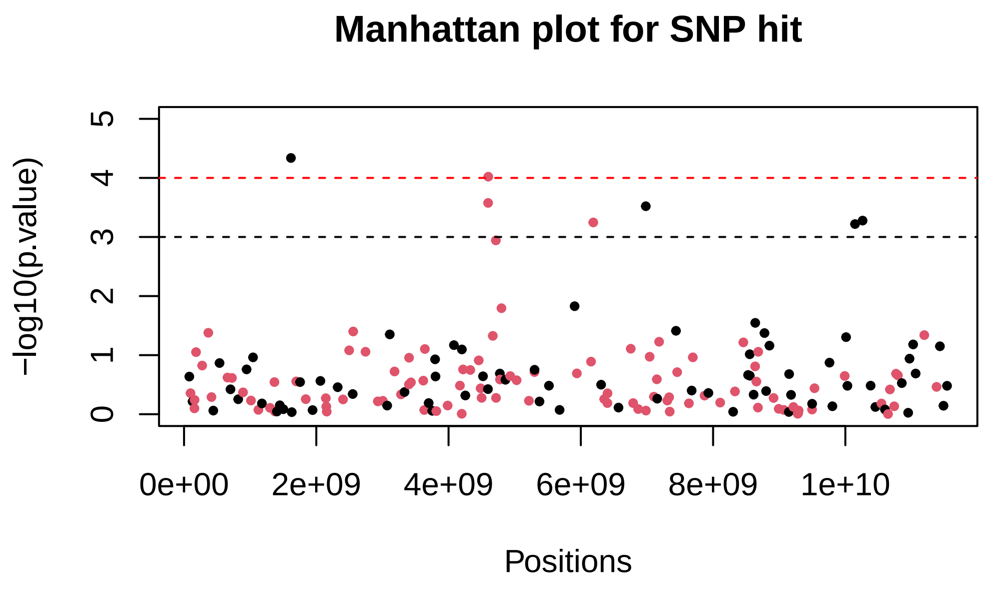
Problem 2.e (8 points)

- Build a weighted genetic risk score that includes all `SNP`s with p-value $< 10^{-4}$, a score with all `SNP`s with p-value $< 10^{-3}$, and a score that only includes `SNP`s on the `FTO` gene
- ***Hint: ensure that the ordering of SNPs is respected.***
- Add the three scores as columns to the `gdm.dt` data table.
- Fit the three scores in separate logistic regression models to test their association with gestational diabetes.
- Report odds ratio, 95% confidence interval and p-value using `kable()` for each score.

```r
snp.hit <- snp.hit %>%
.[, pos := as.numeric(pos)] %>%
.[, cum.pos := cumsum(pos)] %>%
.[, chrom := as.numeric(chrom)]
```

```
Warning in eval(jsub, SDenv, parent.frame()): NAs introduced by coercion
```

```r
chrom.cols <- 1 + snp.hit$chrom %% 2 # 1 for even, 2 for odd chromosomes
with(snp.hit, plot(cum.pos, -log10(p.value), col = chrom.cols, pch = 20,
cex = 0.8, main = "Manhattan plot for SNP hit", xlab = "Positions", ylim = c(0,5)))
abline(h = -log10(1e-4), lty = 2, col = "red")
abline(h = -log10(1e-3), lty = 2, col = "black")
```



To visualise the GWAS, the Manhattan plot is presented below with the hit SNPs are highlighted. This assists in understanding the study results and how the SNP p-values are distributed across the chromosomes, with spikes in certain SNPs that are close in position and have a higher chance of being transmitted together. We have plotted the $-\log(p-values)$ for each SNP with a threshold line at $-\log(1e-4)$ to visualise the hits. We can see that some SNPs close to the hits on the chromosome are approaching "hit" status, but have not quite reached the threshold; which is an example of (not quite significant) linkage disequilibrium. Building weighted genetic risk scores for the different criteria will be performed.

```r
## Function to calculate risk scores, run regression, and store output
weighted.risk <- function(snps.input.dt, criteria.name, data.set){
## Subset data set by SNP of interest
  data.set.grs <- data.set[, .SD, .SDcols = snps.input.dt$SNP]
  ## Risk score weighted by regression coefficient

  weighted.score <- as.vector(as.matrix(data.set.grs) %*%
  snps.input.dt$coefficients)
  ## Add score columns to original data set
  data.set <- cbind(data.set, weighted.score = weighted.score)
  ## Logistic regression model and output, with CI
  mod.weighted <- glm(pheno ~ weighted.score,
                      data=data.set, family="binomial")
  model.sum <- coef(summary(mod.weighted))
  ci.95 <- exp(confint(mod.weighted))[2,]
  output <- data.table(exp(model.sum[2,1]), ci.95[1], ci.95[2], model.sum[2,4])
  output <- cbind(criteria.name,output)
  setnames(data.set, "weighted.score", criteria.name)
  return(list(output = output,
              data.set = data.set,
              mod.weighted = mod.weighted))
}
```

```
## Names of SNPs with pvalue < 1e-4, <1e-3 and SNPs with allele on FTO gene
snp.grs.e4 <- new.gdm.dt[p.value < 1e-4]
snp.grs.e3 <- new.gdm.dt[p.value < 1e-3]
snp.grs.fto <- snp.hit[gene %in% "FTO"]
## Run function for all three criteria
suppressMessages(invisible({
gdm.dt.imputed.score <- copy(gdm.dt.imputed)
e4 <- weighted.risk(snp.grs.e4, "p-value 1e-4", gdm.dt.imputed.score)
gdm.dt.imputed.score <- e4[[2]]
e3 <- weighted.risk(snp.grs.e3, "p-value 1e-3", gdm.dt.imputed.score)
gdm.dt.imputed.score <- e3[[2]]
fto <- weighted.risk(snp.grs.fto, "FTO gene", gdm.dt.imputed.score)
gdm.dt.imputed.score <- fto[[2]]
}))
## Combine information into table
weighted.risk.table <- rbind(e4[[1]], e3[[1]], fto[[1]])
colnames(weighted.risk.table) <- c("Criteria Group","Odds Ratio",
                                   "95% lower","95% Upper","p.value")
weighted.risk.table$p.value <- format(weighted.risk.table$p.value, digits = 4, nsmall = 0)
kable(weighted.risk.table, digits=3,
      caption = "Weighted Genetic Risk Score Model by Criteria") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 16: Weighted Genetic Risk Score Model by Criteria

Criteria Group	Odds Ratio	95% lower	95% Upper	p.value
p-value 1e-4	2.729	1.924	3.911	2.759e-08
p-value 1e-3	1.452	1.281	1.651	7.814e-09
FTO gene	1.414	0.819	2.453	2.152e-01

Problem 2.f (4 points)

- File `GDM.test.txt` (available from the accompanying zip folder on Learn) contains genotypes of another 40 pregnant women with and without gestational diabetes (assume that the reference allele is the same one that was specified in file `GDM.raw.txt`).
- Read the file into variable `gdm.test`.
- For the set of patients in `gdm.test`, compute the three genetic risk scores as defined in **problem 2.e** using the same set of SNPs and corresponding weights.
- Add the three scores as columns to `gdm.test` ***(hint: use the same columnnames as before).***

```r
gdm.test.dt <- fread("data_assignment2/GDM.test.txt")

colnames(gdm.test.dt) <- colnames(gdm.dt)
## Apply weighted risk function to new test set, and add to gdm.test.dt
suppressMessages(invisible({
  e4.test <- weighted.risk(snp.grs.e4, "p.value 1e-4", gdm.test.dt)
  gdm.test.dt <- e4.test[[2]]
  e3.test <- weighted.risk(snp.grs.e3, "p.value 1e-3", gdm.test.dt)
  gdm.test.dt <- e3.test[[2]]
  fto.test <- weighted.risk(snp.grs.fto, "FTO Gene", gdm.test.dt)
  gdm.test.dt <- fto.test[[2]]
}))

weighted.risk.table.test <- rbind(e4.test[[1]], e3.test[[1]], fto.test[[1]])
colnames(weighted.risk.table.test) <- c("Criteria Group","Odds Ratio",
                                        "95% lower","95% Upper","p.value")
weighted.risk.table.test$p.value <- format(weighted.risk.table.test$p.value,
                                           digits = 4, nsmall = 0)
kable(weighted.risk.table.test, digits=3,
      caption = "Weighted Genetic Risk Score Test Model by Criteria") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 17: Weighted Genetic Risk Score Test Model by Criteria

Criteria Group	Odds Ratio	95% lower	95% Upper	p.value
p.value 1e-4	10.650	1.951	100.976	0.01539
p.value 1e-3	2.220	1.259	4.683	0.01410
FTO Gene	1.962	0.152	26.518	0.60095

For sake of ease, we first write a function, `weight.risk()` that will provide a model summary in the form asked for (odds ratios, CIs and p-values). We then build a model for each weighted genetic risk score, and collect the results of this using that function. These are then row binded together to provide a table summary as above.

When calculating weighted scores for all SNPs with p-values less than $1e-4$ or $1e-3$, its interesting to see that you get highly significant odds ratios. Results suggest that weighted scores for SNPs with p-values less than $1e-4$ have an odds ratio point estimate of 2.729, suggesting women with all of these SNPs have a 172.9% higher odds ratio of getting gestational diabetes than not. This ranges between 92.4% and 291.1% across the 95% confidence interval. Likewise for SNPs with p-values less than $1e-3$, there is a 45.2% higher odds ratio of getting gestational diabetes than not. Conversely, for the weighted score for SNPs on the FTO gene you get a p-value of 0.215. Given the confidence interval for the odds-ratio spans over 1, $(0.819-2.453)$, there is no statistical evidence to suggest that the odds-ratio differs from 1, which is when there is an even odds-ratio of having gestational diabetes than not.

Problem 2.g (4 points)

- Use the logistic regression models fitted in **problem 2.e** to predict the outcome of patients in `gdm.test`.
- Compute the test log-likelihood for the predicted probabilities from the three genetic risk score models.

```r
## Isolate the model objects that were output from the function
e4.mod <- e4[[3]]
e3.mod <- e3[[3]]
fto.mod <- fto[[3]]
## Predict patient outcomes for the test data, using the models for the weighted risk
test.data.e4 <- data.frame(weighted.score = gdm.test.dt$`p.value 1e-4`)
pred.e4.mod <- predict(e4.mod, newdata = test.data.e4, type = "response")
test.data.e3 <- data.frame(weighted.score = gdm.test.dt$`p.value 1e-3`)
pred.e3.mod <- predict(e3.mod, newdata = test.data.e3, type = "response")
test.data.fto <- data.frame(weighted.score = gdm.test.dt$`FTO Gene`)
pred.fto.mod <- predict(fto.mod, newdata = test.data.fto, type = "response")
## Predictions into table format
pred.table.test <- cbind(gdm.test.dt$ID, pred.e4.mod, pred.e3.mod, pred.fto.mod)
colnames(pred.table.test) <- c("Patient","p.value 1e-4","p.value 1e-3","FTO Gene")

kable(head(pred.table.test), caption = "Log-likelihoods for Predicted Probabilities") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

```r
## Compute test log-likelihoods
e4.log.lik <- sum(dbinom(gdm.test.dt$pheno, prob=pred.e4.mod, size=1, log=TRUE))
```

Table 18: Log-likelihoods for Predicted Probabilities

Patient	p.value 1e-4	p.value 1e-3	FTO Gene
1101	0.4940467	0.4056544	0.5130166
1102	0.3380768	0.3491907	0.5538554
1104	0.5524594	0.5016399	0.5130166
1105	0.4425963	0.4607599	0.5538554
1106	0.5524594	0.5016399	0.5130166
1107	0.3380768	0.3491907	0.5130166

```r
e3.log.lik <- sum(dbinom(gdm.test.dt$pheno, prob=pred.e3.mod, size=1, log=TRUE))
fto.log.lik <- sum(dbinom(gdm.test.dt$pheno, prob=pred.fto.mod, size=1, log=TRUE))
## Combine into table
log.lik.table <- cbind(e4.log.lik, e3.log.lik, fto.log.lik)
colnames(log.lik.table) <- c("P-Value 10^-4","P-Value 10^-3","FTO Gene")
kable(log.lik.table, caption = "Log-likelihoods for Predicted Probabilities") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 19: Log-likelihoods for Predicted Probabilities

P-Value 10^-4	P-Value 10^-3	FTO Gene
-25.06824	-24.77693	-28.05355

Comparison of the models based on log-likelihoods indicates that the weighted risk score model under the 40 criteria of p-value $< 1e-3$ is a better fit to the data set, since a higher value indicates a generally better model fit. Additional predictors in a model will almost always increase the log-likelihood, however this is not an issue here since all three models are based on the weighted genetic risk score variable.

Given we have a binary outcome, our outcome variable can be modelled as a Bernoulli distribution with probabilities equivalent to our predicted test values (as these range between 0 and 1 and reflect the risk of having gestational diabetes). To calculate our log-likelihood values, we could either calculate this manually using the Bernoulli distributions probability mass function and calculating the likelihood and corresponding log-likelihood functions, or alternatively we can pass this through the `dbinom` function in `R` with a size of 1. By setting log=TRUE, we are telling `R` that we want this to be on the log scale.

By looking at all log-likelihoods across all predictions, can see that the model with SNPs with p-values less than 10^{-3} performs best, given this has the largest log-likelihood out of all the models. A high log-likelihood suggests a better fit of the model to the data, which in this case reflects good predictive performance.

Problem 2.h (4points)

- File `GDM.study2.txt` (available from the accompanying zip folder on Learn) contains the summary statistics from a different study on the same set of SNPs.
- Perform a meta-analysis with the results obtained in **problem 2.c** ***(hint: remember that the effect alleles should correspond)*** - produce a summary of the meta-analysis results for the set of SNPs with meta-analysis p-value $< 10^{-4}$ sorted by increasing p-value.

```r
gdm2.dt <- fread("data_assignment2/GDM.study2.txt")

# all SNP identifiers are the same
# Creating new columns in gdm2.dt which is snp with effect allele
gdm2.dt[, SNP := paste(snp, effect.allele, sep = "_")]
# Check if both studies contain the same SNP with alleles (only effect)
match.snp.gdm <- gdm2.dt$SNP %in% new.gdm.dt$SNP
match.snp.origin <- new.gdm.dt$SNP %in% gdm2.dt$SNP
sum.match.gdm <- sum(match.snp.gdm)
sum.match.origin <- sum(match.snp.origin)
```

```r
## Only keep SNPs which match across both studies (effect allele)
gdm2.dt.meta <- gdm2.dt[match.snp.gdm]
snp.model.meta <- new.gdm.dt[match.snp.origin]
## Add new column with study names
snp.model.meta$study <- "Original Study"
gdm2.dt.meta$study <- "GDM Study"
## Calculate p-values for gdm.dt data
gdm2.dt.meta[, p.value := pnorm(-abs(beta)/se)*2]
## Only obtain SNPs with p-values < 10^{-4} for original model results
high.sig.snp.results <- snp.model.meta[p.value < 1e-04,]
## Only wanting to keep Study, SNPName, SNPCoef, StandardError, and pvalue
high.sig.snp.results <- high.sig.snp.results[, c("study","SNP", "coefficients",
                                                "std.error","p.value")]
## Only obtain SNPs with p-values < 10^{-4} for original model results
high.sig.gdm.results <- gdm2.dt.meta[p.value < 1e-04,]
## Only wanting to keep Study, SNPName, beta, se, and pvalue
high.sig.gdm.results <- high.sig.gdm.results[, c("study","SNP",
                                                "beta","se", "p.value")]

## Columns are in all the same position, binding significant results together
sig.results <- rbind(high.sig.snp.results, high.sig.gdm.results, use.names=FALSE)
## Making sure results for SNPs are easily comparable
```

```r
setorder(sig.results, SNP)
sig.results$p.value <- format(sig.results$p.value, digits = 4, nsmall = 0)
kable(sig.results,
caption = "Summary of meta-analysis results for SNPs with p-values $<1e-4$") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 20: Summary of meta-analysis results for SNPs with p-values $< 1e - 4$

study	SNP	coefficients	std.error	p.value
Original Study	rs12243326_A	0.6454198	0.1583787	4.598e-05
GDM Study	rs12243326_A	1.1468970	0.1610925	1.083e-12
Original Study	rs2237897_T	-0.4394456	0.1126133	9.530e-05
GDM Study	rs2237897_T	-1.1691220	0.2205240	1.148e-07

Problem 3 (33 points)

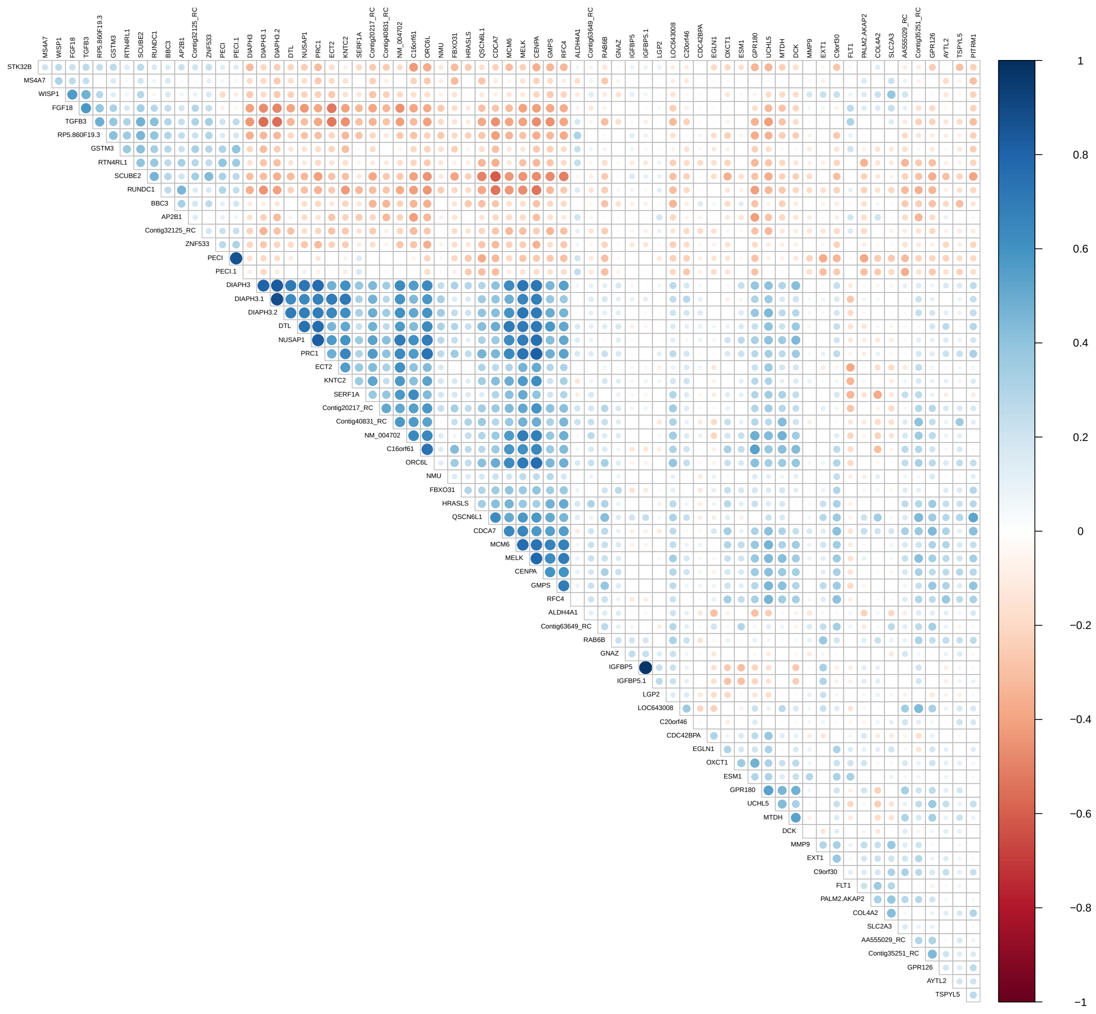
File `nki.csv` (available from the accompanying zip folder on Learn) contains data for 144 breast cancer patients. The dataset contains a binary outcome variable (`Event`, indicating the insurgence of further complications after operation), covariates describing the tumour and the age of the patient, and gene expressions for 70 genes found to be prognostic of survival.

Problem 3.a (6 points)

- Compute the correlation matrix between the gene expression variables, and display it so that a block structure is highlighted using the `corrplot` package.
- Discuss what you observe.
- Identify the unique pairs of (distinct) variables that have correlation coefficient greater than 0.80 in absolute value and report their correlation coefficients.

```r
cancer.dt <- fread("data_assignment2/nki.csv")
## Identify factor variables and factor outcome
factor.vars <- c("Event", "Diam", "LymphNodes", "EstrogenReceptor","Grade")
cancer.dt[, (factor.vars) := lapply(.SD, function(x) as.factor(x)),
          .SDcols = factor.vars]
## Isolate gene names and find correlations
gene.cols <- colnames(cancer.dt[,-c(1:6)])
cor.cancer <- cancer.dt[, ..gene.cols] %>% #subset of numeric columns
                cor(use="pairwise.complete")
## Highlight block structure by ordering into correlation clusters
corrplot(cor.cancer, order = "hclust", diag = FALSE,
          tl.col = "black", tl.cex = 0.5, mar = c(1,1,1,1), type = 'upper',
          title = "Correlation matrix (ordered by hierarchical clustering)")
```

Correlation matrix (ordered by hierarchical clustering)



```r
## Set upper triangular part of matrix to NA and disregard (incl. diagonal)
cor.cancer.high <- cor.cancer
cor.cancer.high[lower.tri(cor.cancer.high)] <- NA
diag(cor.cancer.high) <- NA

## Select correlations with abs value greater than 0.8 and set names
high.corr <- subset(as.data.frame.table(cor.cancer.high), abs(Freq) > 0.8)
rownames(high.corr) <- NULL
colnames(high.corr) <- c("Covariate 1","Covariate 2", "Correlation Coefficient")
```

```r
kable(high.corr,  digits = 3, caption = "Unique Pairs with High Correlation") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 21: Unique Pairs with High Correlation

Covariate 1	Covariate 2	Correlation Coefficient
DIAPH3	DIAPH3.1	0.803
DIAPH3	DIAPH3.2	0.834
DIAPH3.1	DIAPH3.2	0.887
PECI	PECI.1	0.870
IGFBP5	IGFBP5.1	0.978
NUSAP1	PRC1	0.830
PRC1	CENPA	0.818

From the correlation plot above, it can be seen that there are strong correlations between certain gene expressions. The clusters of blue indicate concentrations of highly positively correlated genes, while the clusters of red indicate concentrations of negatively correlated genes. During principal component analysis (PCA), this large set of correlated variables will be summarized into smaller axes of variation. In the analysis below, it is found that there are seven distinct pairs of variables that have a very high positive correlations, above 0.8. No variable pairs were found to have a very strong negative correlation, -0.8.

Problem 3.b (8 points)

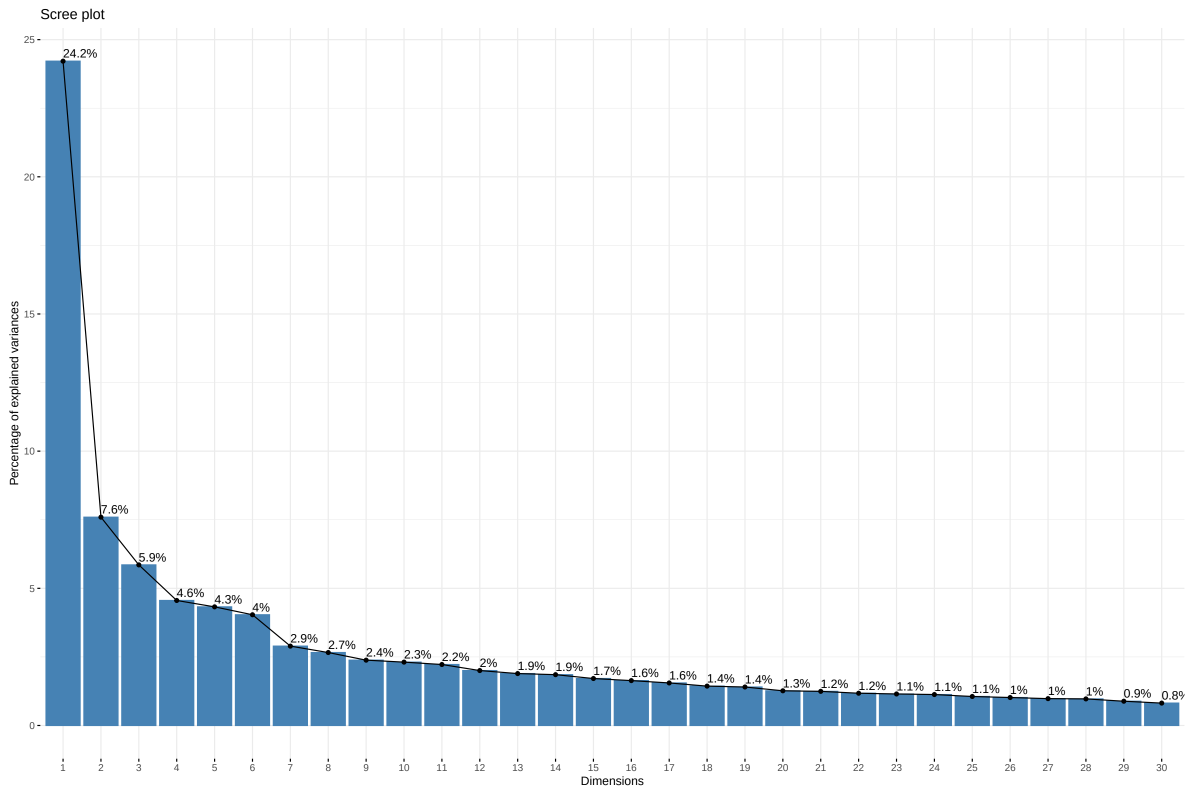
- Perform PCA analysis (only over the columns containing gene expressions) in order to derive a patient-wise summary of all gene expressions (dimensionality reduction).
- Decide which components to keep and justify your decision.
- Test if those principal components are associated with the outcome in unadjusted logistic regression models and in models adjusted for `age`, `estrogen receptor` and `grade`.
- Justify the difference in results between unadjusted and adjusted models.

```r
## Run PCA on numeric covariate columns for gene expressions
pca.vars <- prcomp(cancer.dt[, ..gene.cols], center = T, scale = T)
summary(pca.vars)
```

```
Importance of components:
                          PC1     PC2     PC3     PC4     PC5     PC6     PC7
Standard deviation     4.1171 2.30541 2.02437 1.78597 1.73982 1.68091 1.42309
Proportion of Variance 0.2422 0.07593 0.05854 0.04557 0.04324 0.04036 0.02893
Cumulative Proportion  0.2422 0.31808 0.37662 0.42219 0.46543 0.50580 0.53473
                           PC8     PC9    PC10    PC11    PC12    PC13    PC14
```

```
Standard deviation     1.36441 1.29119 1.2715 1.24741 1.18388 1.15101 1.13883
Proportion of Variance 0.02659 0.02382 0.0231 0.02223 0.02002 0.01893 0.01853
Cumulative Proportion  0.56132 0.58514 0.6082 0.63046 0.65049 0.66941 0.68794
                          PC15    PC16    PC17    PC18    PC19    PC20    PC21
Standard deviation     1.09473 1.07016 1.04187 1.00234 0.99086 0.94095 0.93322
Proportion of Variance 0.01712 0.01636 0.01551 0.01435 0.01403 0.01265 0.01244
Cumulative Proportion  0.70506 0.72142 0.73693 0.75128 0.76531 0.77796 0.79040
                          PC22    PC23    PC24    PC25    PC26    PC27    PC28
Standard deviation     0.90727 0.89675 0.88859 0.86019 0.84462 0.82782 0.82368
Proportion of Variance 0.01176 0.01149 0.01128 0.01057 0.01019 0.00979 0.00969
Cumulative Proportion  0.80216 0.81364 0.82492 0.83549 0.84569 0.85548 0.86517
                          PC29    PC30    PC31    PC32    PC33    PC34   PC35
Standard deviation     0.78694 0.75594 0.73942 0.70569 0.69414 0.67129 0.6639
Proportion of Variance 0.00885 0.00816 0.00781 0.00711 0.00688 0.00644 0.0063
Cumulative Proportion  0.87401 0.88218 0.88999 0.89710 0.90399 0.91042 0.9167
                          PC36    PC37    PC38    PC39    PC40    PC41    PC42
Standard deviation     0.63815 0.61964 0.59947 0.58447 0.57195 0.55097 0.53820
Proportion of Variance 0.00582 0.00549 0.00513 0.00488 0.00467 0.00434 0.00414
Cumulative Proportion  0.92254 0.92802 0.93316 0.93804 0.94271 0.94705 0.95118
                          PC43    PC44    PC45    PC46    PC47    PC48    PC49
Standard deviation     0.52029 0.51211 0.49533 0.48712 0.47079 0.44565 0.41879
Proportion of Variance 0.00387 0.00375 0.00351 0.00339 0.00317 0.00284 0.00251
Cumulative Proportion  0.95505 0.95880 0.96230 0.96569 0.96886 0.97170 0.97420
                          PC50    PC51   PC52    PC53    PC54    PC55    PC56
Standard deviation     0.40556 0.39328 0.3925 0.38502 0.36669 0.36205 0.33734
Proportion of Variance 0.00235 0.00221 0.0022 0.00212 0.00192 0.00187 0.00163
Cumulative Proportion  0.97655 0.97876 0.9810 0.98308 0.98500 0.98687 0.98850
                          PC57    PC58    PC59    PC60    PC61    PC62    PC63
Standard deviation     0.32150 0.30744 0.28898 0.28186 0.27274 0.25622 0.24118
Proportion of Variance 0.00148 0.00135 0.00119 0.00113 0.00106 0.00094 0.00083
Cumulative Proportion  0.98998 0.99133 0.99252 0.99365 0.99472 0.99565 0.99649
                          PC64    PC65    PC66    PC67    PC68   PC69    PC70
Standard deviation     0.23024 0.21442 0.19886 0.19371 0.17927 0.1677 0.09833
Proportion of Variance 0.00076 0.00066 0.00056 0.00054 0.00046 0.0004 0.00014
Cumulative Proportion  0.99724 0.99790 0.99846 0.99900 0.99946 0.9999 1.00000
```

```r
## Scree Plot for each Principal Component
fviz_eig(pca.vars, addlabels = TRUE, ncp = 30)
```



The PCA process has resulted in 70 principal components, in order of highest standard deviation to lowest. A scree plot is used to visualize the variance explained by the principal components. The scree plot indicates that after 6^{th} variable, the curve begins to flatten but not drastically. More variation can be captured by including more components and the cut-off is not very distinct.

To assist in deciding which principal components to include in the model, the variability captured by the components is assessed. In order to have 80% of the variability, 22 principal components would need to be included in the model. While this is a rather complex model, including less than 10 components would result in only 60% of the variability being captured.

```r
## Compute amount of variability of components - sqrt of eigenvalues stored in sdev
sdev <- pca.vars$sdev^2
perc.expl <- sdev / sum(sdev)
## Calculate number of components needed to capture 80% of the variability
variability <- data.frame(variability = sort(perc.expl, decreasing=TRUE))
variability$cumulative <- cumsum(variability$variability)
num.comp.var.80 <- which.min(abs(variability$cumulative - 0.8))
num.comp.var.70 <- which.min(abs(variability$cumulative - 0.7))
num.comp.var.60 <- which.min(abs(variability$cumulative - 0.6))
num.comp.var.1se <- which.min(abs(pca.vars$sdev-1.0))
var.capture <- data.table("no. PCs",num.comp.var.80, num.comp.var.70,
                          num.comp.var.60, num.comp.var.1se)
colnames(var.capture) <- c("","80%", "70%", "60%", "1 std.error")
kable(var.capture, caption = "Number of principal components needed for explainability") |
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 22: Number of principal components needed for explainability

	80%	70%	60%	1 std.error
no. PCs	22	15	10	18

```r
## Select number of PCs to be checked in glm models
pca.for.model. <- pca.vars$x[, 1:num.comp.var.1se]
pca.for.model <- colnames(pca.for.model.)
## Check each PC for significance in a glm models
adj.sig.pc <- unadj.sig.pc <- c()
for (i in pca.for.model){
  unadj.log.model <- glm(Event ~ pca.vars$x[,i], data = cancer.dt, family="binomial")
  unadj.p.value <- coef(summary(unadj.log.model))[2,4]
  if(unadj.p.value < 0.05){
    unadj.sig.pc <- c(unadj.sig.pc, "Significant")
  }
  else{
    unadj.sig.pc <- c(unadj.sig.pc, "Insignificant")
  }
  adj.log.model <- glm(Event ~ pca.vars$x[,i] + Age + EstrogenReceptor + Grade,
                       data = cancer.dt, family="binomial")
  adj.p.value <- coef(summary(adj.log.model))[2,4]
  if(adj.p.value < 0.05){
    adj.sig.pc <- c(adj.sig.pc, "Significant")
  }
  else{
    adj.sig.pc <- c(adj.sig.pc, "Insignificant")
  }
}
sig.pc <- data.table(colnames(pca.for.model.), adj.sig.pc, unadj.sig.pc)
colnames(sig.pc) <- c("PCs", "Adjusted", "Unadjusted")
kable(list(sig.pc[1:9,], sig.pc[10:18,]),booktabs = TRUE,
  caption = "Significant PCs from Unadjusted and Adjusted Model") |>
```

```r
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 23: Significant PCs from Unadjusted and Adjusted Model

PCs	Adjusted	Unadjusted	PCs	Adjusted	Unadjusted
PC1	Insignificant	Significant	PC10	Insignificant	Insignificant
PC2	Insignificant	Insignificant	PC11	Insignificant	Significant
PC3	Significant	Significant	PC12	Insignificant	Insignificant
PC4	Insignificant	Insignificant	PC13	Insignificant	Insignificant
PC5	Insignificant	Insignificant	PC14	Insignificant	Insignificant
PC6	Insignificant	Insignificant	PC15	Insignificant	Insignificant
PC7	Insignificant	Insignificant	PC16	Insignificant	Insignificant
PC8	Insignificant	Insignificant	PC17	Insignificant	Insignificant
PC9	Insignificant	Insignificant	PC18	Insignificant	Insignificant

```r
## Overall regression on PCs
unadj.log.model <- glm(Event ~ pca.for.model., data = cancer.dt, family="binomial")
summary(unadj.log.model)
```

```

Call:
glm(formula = Event ~ pca.for.model., family = "binomial", data = cancer.dt)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8953  -0.7753  -0.4492   0.8509   2.3198

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)        -0.957440   0.223917  -4.276  1.9e-05 ***
pca.for.model.PC1   0.163396   0.056265   2.904  0.00368 **
pca.for.model.PC2  -0.083883   0.091036  -0.921  0.35683
pca.for.model.PC3   0.329095   0.113730   2.894  0.00381 **
pca.for.model.PC4  -0.205872   0.118185  -1.742  0.08152 .
pca.for.model.PC5  -0.067448   0.120020  -0.562  0.57413
pca.for.model.PC6   0.186068   0.132370   1.406  0.15982
pca.for.model.PC7  -0.167816   0.151484  -1.108  0.26794
pca.for.model.PC8   0.226880   0.160300   1.415  0.15697
pca.for.model.PC9  -0.038896   0.156144  -0.249  0.80328
pca.for.model.PC10 -0.221624   0.166930  -1.328  0.18430
pca.for.model.PC11 -0.397418   0.181944  -2.184  0.02894 *
pca.for.model.PC12  0.055617   0.175517   0.317  0.75134
pca.for.model.PC13 -0.008423   0.183102  -0.046  0.96331
pca.for.model.PC14 -0.170226   0.183256  -0.929  0.35294
pca.for.model.PC15 -0.050842   0.194379  -0.262  0.79366
pca.for.model.PC16 -0.123973   0.190301  -0.651  0.51475
pca.for.model.PC17 -0.457203   0.213087  -2.146  0.03190 *
pca.for.model.PC18 -0.353909   0.217057  -1.630  0.10300
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 183.32  on 143  degrees of freedom
Residual deviance: 142.82  on 125  degrees of freedom
AIC: 180.82

Number of Fisher Scoring iterations: 5
```

```r
adj.log.model <- glm(Event ~ pca.for.model. + Age + EstrogenReceptor + Grade,
                      data = cancer.dt, family="binomial")
summary(adj.log.model)
```

```
Call:
glm(formula = Event ~ pca.for.model. + Age + EstrogenReceptor +
    Grade, family = "binomial", data = cancer.dt)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5704  -0.7743  -0.4403   0.7118   2.4782

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)               2.541174   2.145756   1.184   0.2363
pca.for.model.PC1         0.062986   0.090996   0.692   0.4888
pca.for.model.PC2         0.030305   0.127440   0.238   0.8120
pca.for.model.PC3         0.299268   0.117587   2.545   0.0109 *
pca.for.model.PC4        -0.200687   0.124503  -1.612   0.1070
pca.for.model.PC5        -0.045292   0.123251  -0.367   0.7133
pca.for.model.PC6         0.268410   0.156133   1.719   0.0856 .
pca.for.model.PC7        -0.181190   0.156285  -1.159   0.2463
pca.for.model.PC8         0.333023   0.181136   1.839   0.0660 .
pca.for.model.PC9        -0.001776   0.162610  -0.011   0.9913
pca.for.model.PC10       -0.247489   0.168290  -1.471   0.1414
pca.for.model.PC11       -0.353997   0.187265  -1.890   0.0587 .
pca.for.model.PC12        0.036663   0.181329   0.202   0.8398
pca.for.model.PC13        0.087377   0.199131   0.439   0.6608
pca.for.model.PC14       -0.199706   0.194082  -1.029   0.3035
pca.for.model.PC15       -0.090025   0.205436  -0.438   0.6612
pca.for.model.PC16       -0.095292   0.194560  -0.490   0.6243
pca.for.model.PC17       -0.354552   0.219782  -1.613   0.1067
pca.for.model.PC18       -0.314638   0.215769  -1.458   0.1448
Age                      -0.060114   0.045241  -1.329   0.1839
EstrogenReceptorPositive -0.981802   1.052608  -0.933   0.3510
GradePoorly diff          0.320094   0.544208   0.588   0.5564
GradeWell diff           -0.551021   0.614747  -0.896   0.3701
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 183.32  on 143  degrees of freedom
Residual deviance: 138.59  on 121  degrees of freedom
AIC: 184.59

Number of Fisher Scoring iterations: 5
```

```r
## Perform likelihood ratio test
p.value <- pchisq(unadj.log.model$deviance - adj.log.model$deviance,
                    df = 4, lower.tail=FALSE)
cat("Likelihood ratio test p-value:", p.value, "\n")
```

```
Likelihood ratio test p-value: 0.3759545
```

```r
## Confirm likelihood ratio test
lrtest(unadj.log.model, adj.log.model)
```

```
Likelihood ratio test

Model 1: Event ~ pca.for.model.
Model 2: Event ~ pca.for.model. + Age + EstrogenReceptor + Grade
  #Df  LogLik Df  Chisq Pr(>Chisq)
1  19 -71.411
2  23 -69.296  4 4.2286      0.376
```

Regression is run on the individual components to determine which of the 18 are significant in association with breast cancer patients having a post-surgery event. Each component is run in a unadjusted logistic regression model, and an adjusted model that includes the additional variables of `age`, `estrogen receptor`, and `grade`.

When run individually, the unadjusted regression models indicate that principal components PC 1, PC 3, PC 11 and PC 17 are all significant. The results for the adjusted model are different from the unadjusted model because the additional parameters of `age`, `estrogen receptor`, and `grade` affect the model, increasing the PC p-values. Overall regression model fits with the 18 principal components produce slightly different results, indicating that PC 17 is also significant.

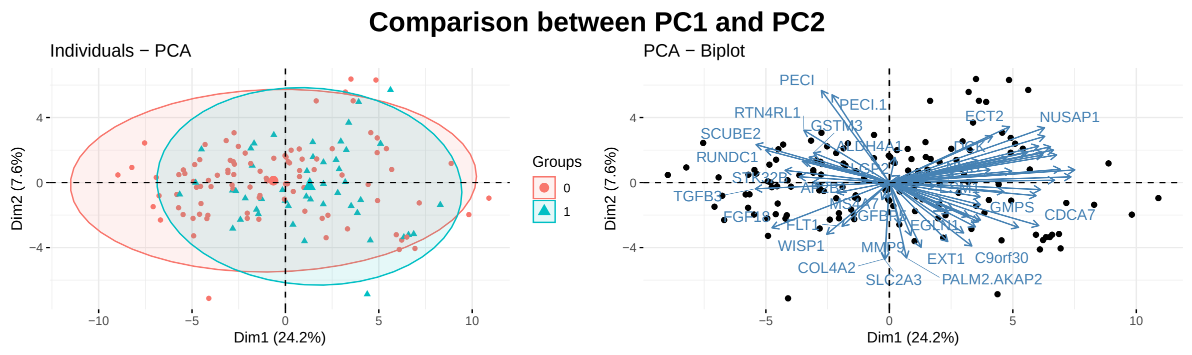
A likelihood ratio test can be performed on these two nested modesl. This test is performed by calculating the difference in deviances of the two models, which follows a χ^2-distribution with degrees of freedom equal to the difference in number of parameters, four in this case since grade has three levels. With a p-value of 0.376, the results of likelihood ratio test show that there is not evidence to add the additional parameters for the adjusted model. Therefore this further suggests that the addition of these parameters is diluting the model, making it more complex and affecting the significance of the principal component covariates.

Problem 3.c (8 points)

- Use PCA plots to compare the main drivers with the correlation structure observed in **problem 3.a**.
- Examine how well the dataset may explain your outcome.
- Discuss your findings in full details and suggest any further steps if needed.

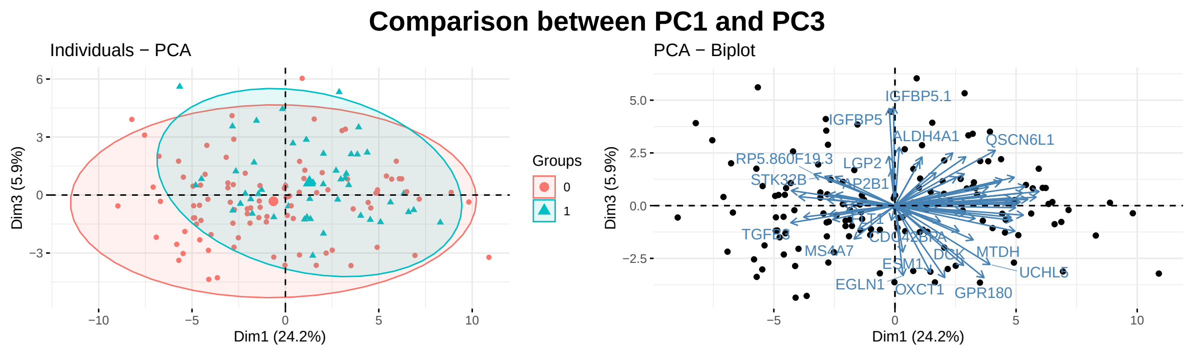
```r
axes <- c(1,2)
p1 <- fviz_pca_ind(pca.vars, geom='point', axes = axes,
habillage = cancer.dt$Event,
addEllipses = T)
p2 <- fviz_pca_biplot(pca.vars, geom='point', repel = T,axes = axes)
plot <- ggpubr::ggarrange(p1,p2)
annotate_figure(plot, top = text_grob("Comparison between PC1 and PC2",
                color = "black", face = "bold", size = 20))
```

```
Warning: ggrepel: 40 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



```
axes <- c(1,3)
p1 <- fviz_pca_ind(pca.vars, geom='point', axes = axes,
habillage = cancer.dt$Event,
addEllipses = T)
p2 <- fviz_pca_biplot(pca.vars, geom='point', repel = T,axes = axes)
plot <- ggpubr::ggarrange(p1,p2)
annotate_figure(plot, top = text_grob("Comparison between PC1 and PC3",
                  color = "black", face = "bold", size = 20))
```

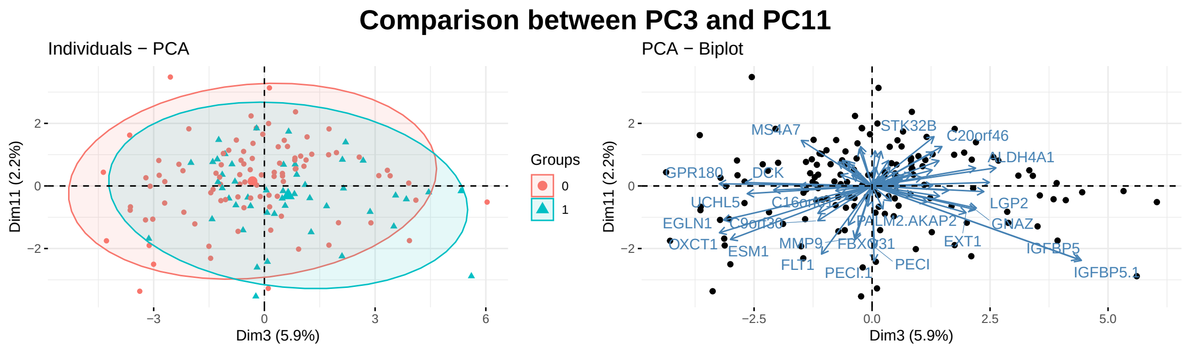
```
Warning: ggrepel: 51 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



```
axes <- c(3,11)
p1 <- fviz_pca_ind(pca.vars, geom='point', axes = axes,
habillage = cancer.dt$Event,
```

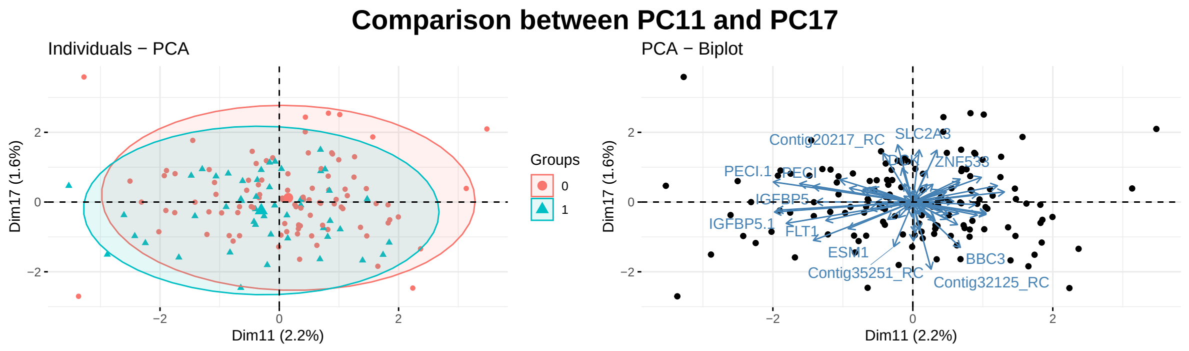
```r
addEllipses = T)
p2 <- fviz_pca_biplot(pca.vars, geom='point', repel = T,axes = axes)
plot <- ggpubr::ggarrange(p1,p2)
annotate_figure(plot, top = text_grob("Comparison between PC3 and PC11",
                  color = "black", face = "bold", size = 20))
```

```
Warning: ggrepel: 47 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



```r
axes <- c(11,17)
p1 <- fviz_pca_ind(pca.vars, geom='point', axes = axes,
habillage = cancer.dt$Event,
addEllipses = T)
p2 <- fviz_pca_biplot(pca.vars, geom='point', repel = T,axes = axes)
plot <- ggpubr::ggarrange(p1,p2)
annotate_figure(plot, top = text_grob("Comparison between PC11 and PC17",
                  color = "black", face = "bold", size = 20))
```

```
Warning: ggrepel: 57 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



Plotting two principal components against each other can help identify how well they separate the data from having an event after surgery or not having an event. The data points of the outcomes are shown below, against a selection of two principal components. The ellipses of 0 and 1 show the outcome of an event occurring, and contain 95% of the data in each group, with the centroid of the group shown as an enlarged data point.

In this data set, there are many variables that are associated with the outcome, hence the large number of principal components needed to capture the variability. However, it can be seen that there is still quite a bit of separating power in a few PCA components, evidenced by the separation between the ellipses in the plots below. Different combinations of principal components provide different separation power, and plots with overlapping ellipses indicate that the two components are not capturing different variation, thus including both in a model is probably not optimal.

The first two components contain the largest variation in the data, and PC 1 and PC 2 show quite a bit of distinction in the ellipses. However, examination of the biplot (which shows the linear combinations of the components) reveals that the magnitude of the variable arrows is greater along the x-axis, indicating that PC 1 is assigning more weight to these variables. The magnitude of the variable arrows along the y-axis appears to be lower, meaning that PC 2 is assigning less weight to these variables and is capturing less of the variability, which aligns with what was observed from the unadjusted regression model, as PC 2 shows as not significant in the model.

As can be expected, variables that were found to be highly correlated with each other are closely aligned in the biplot, since the principal components are the eigenvectors of the variance-covariance matrix of the the predictor variables. For example, the similar direction and magnitude of `CENPA` and `PRC1`, which have a correlation of 0.818, can be observed in the biplot for PC 1 and PC 2.

Since it appears the significant principal components from the model capture more of the variability, then example plots of these components against each other can be useful in evaluating how the gene expression data is explaining the outcome of event. Example comparisons of PC 1, PC 3, PC 11, and PC 17 are plotted against each other below.

In comparison of PC 1 and PC 3 above, the very strong correlation between `IGFBP5` and `IGFB5.1` of 0.978 can be seen, as the linear combinations for these variables are almost completely aligned. Similarly for the strong correlation between `PECI` and `PECI.1` as well. There is less overlap in the ellipses, indicating the separation power of these components is stronger.

For the significant PCA components from the regression models, since their overall variation is lower than the earlier components (since they are ordered by the size of their eigenvalues), it can be seen that the ellipses are not as distinct. PC 11 and PC 17 below appear to be capturing similar variation in the data, and the magnitude of their variable arrows is quite a bit lower than earlier components.

From the different plots, it is observed that they generally follow what was observed in the correlation plot, which is expected since the PCA components are the eigenvectors of the variance-covariance matrix of the the predictor variables.

This process helps to see the importance of covariate selection and reduction of highly correlated variables in a model. Additionally, it may be helpful to remove some variables that are not correlated with the outcome, before performing PCA. It is important to note that no data cleaning, outlier analysis, or examination of the errors has been performed, which would normally be part of the process.

When observing the plots of these components, it appears the explanatory variables of this data set do provide some separation power and ability to explain the outcome. Since the ellipses overlap quite significantly for some components, the explanatory power is not perfect. However, building models on this data set may be able to provide predictive power for the outcome.

Problem 3.d (11 points)

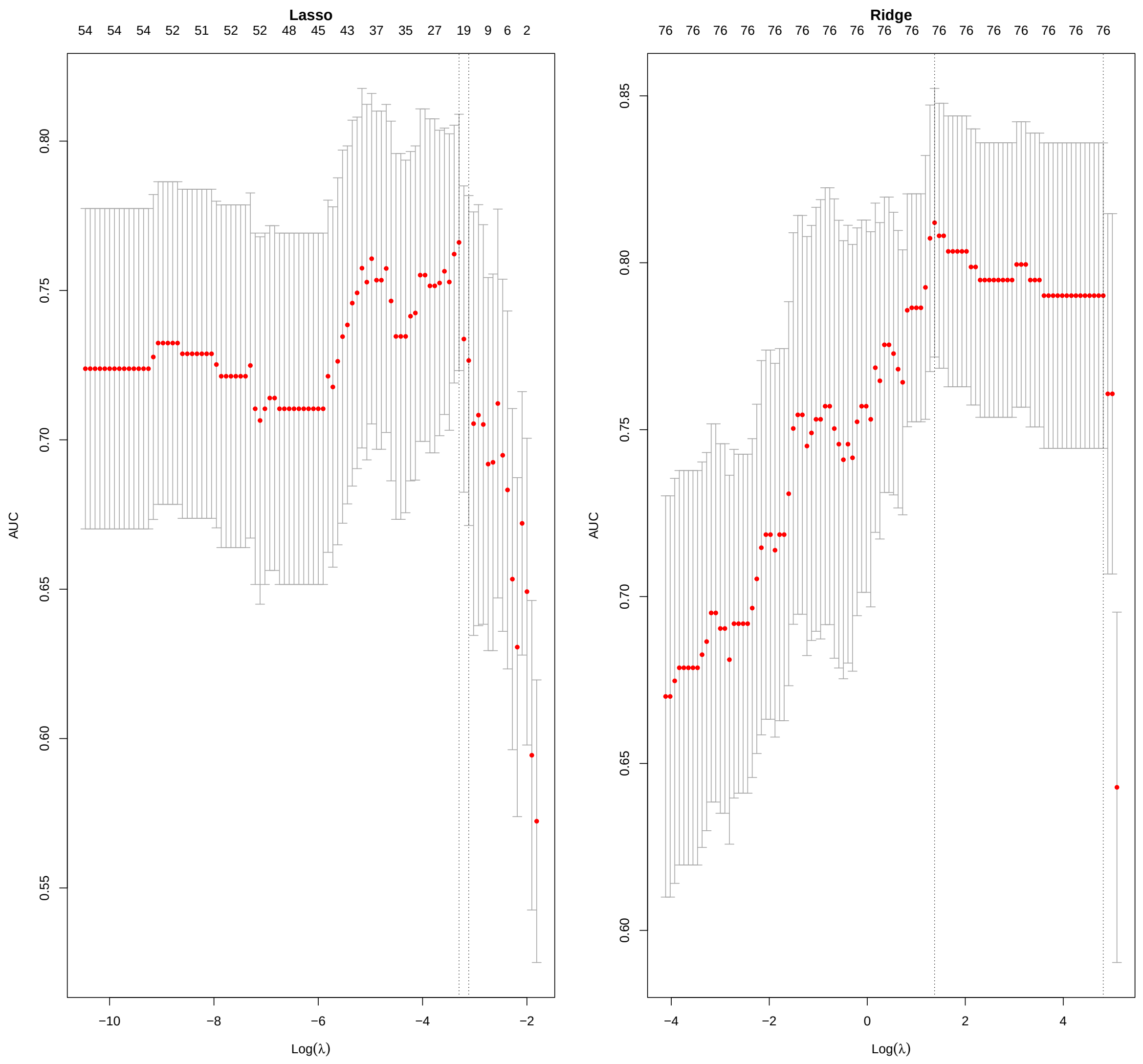
- Based on the models we examined in the labs, fit an appropriate model with the aim to provide the most accurate prognosis you can for patients.
- Discuss and justify your decisions with several experiments and evidences.

```r
## Create the training data 70-30 split
set.seed(984065)
train.idx <-createDataPartition(cancer.dt$Event, p=0.7)$Resample1
## Split into train and test subsets
cancer_train.dt <- cancer.dt[train.idx,]
cancer_test.dt <- cancer.dt[!train.idx,]
## Store the outcome and the covariates separately
y_cancer_train.dt <- cancer_train.dt$Event
x_cancer_train.dt <- model.matrix(Event ~ .,cancer_train.dt)
y_cancer_test.dt <- cancer_test.dt$Event
```

```
x_cancer_test.dt <- model.matrix(Event ~ .,cancer_test.dt)
```

Lasso & Ridge Regression Models.

```
## Fit lasso and ridge models
fit.cv.lasso <- cv.glmnet(x_cancer_train.dt, y_cancer_train.dt, family = "binomial",
type.measure = "auc", alpha = 1)
fit.cv.ridge <- cv.glmnet(x_cancer_train.dt, y_cancer_train.dt, family = "binomial",
type.measure = "auc", alpha = 0)
## Plot the cross-validation curves
par(mfrow=c(1,2), mar=c(4,4,5,2))
plot(fit.cv.lasso, main="Lasso")
plot(fit.cv.ridge, main="Ridge")
```



```r
## Find indices of lambdas that maximizes AUC
idx_lasso.min <- fit.cv.lasso$index["min",]
idx_ridge.min <- fit.cv.ridge$index["min",]

## Reporting model size of each regression
modelsize <- data.table("Model Size",
                        fit.cv.lasso$nzero[idx_lasso.min],
                        fit.cv.ridge$nzero[idx_ridge.min])
colnames(modelsize) <- c("","Lasso", "Ridge")
kable(modelsize,
  caption = "Model Size of Lasso and Ridge Regression") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 24: Model Size of Lasso and Ridge Regression

	Lasso	Ridge
Model Size	22	76

The model size for a ridge regression model requires all 76 covariates for maximum AUC, and this is a complex and undesirable model. Lasso regression only requires 24 covariates, which is the more suitable model, out of these two options.

Logistic Regression

To build a regular logistic regression model for this data, it is difficult due to the fact that the data set contains 75 covariates and only 144 observations. Even backwards elimination and forward selection are not feasible options for covariate selection, as the algorithms will not converge. Therefore it is evident that dimensionality reduction may help in building this model.

Principal Component Analysis

To reduce the dimensions of this data set, PCA is performed on the gene expression covariates of the training data.

```r
## Run PCA on numeric covariate columns for gene expressions
pca.vars <- prcomp(cancer_train.dt[, ..gene.cols], center = T, scale = T)
summary(pca.vars)
```

```
Importance of components:
                          PC1     PC2     PC3     PC4     PC5     PC6     PC7
Standard deviation     4.0080 2.50634 2.06920 1.87886 1.80584 1.63588 1.54130
Proportion of Variance 0.2295 0.08974 0.06117 0.05043 0.04659 0.03823 0.03394
Cumulative Proportion  0.2295 0.31923 0.38039 0.43082 0.47741 0.51564 0.54958
                           PC8     PC9    PC10    PC11    PC12    PC13    PC14
Standard deviation     1.37656 1.34335 1.32861 1.24840 1.18439 1.16273 1.14199
Proportion of Variance 0.02707 0.02578 0.02522 0.02226 0.02004 0.01931 0.01863
Cumulative Proportion  0.57665 0.60243 0.62764 0.64991 0.66995 0.68926 0.70789
                          PC15    PC16    PC17    PC18    PC19    PC20    PC21
Standard deviation     1.12919 1.09268 1.06931 1.03400 1.00361 0.96441 0.94747
```

```
Proportion of Variance 0.01822 0.01706 0.01633 0.01527 0.01439 0.01329 0.01282
Cumulative Proportion  0.72611 0.74316 0.75950 0.77477 0.78916 0.80245 0.81527
                          PC22    PC23    PC24    PC25    PC26    PC27    PC28
Standard deviation     0.93648 0.91312 0.88381 0.88249 0.81876 0.79004 0.77861
Proportion of Variance 0.01253 0.01191 0.01116 0.01113 0.00958 0.00892 0.00866
Cumulative Proportion  0.82780 0.83971 0.85087 0.86200 0.87157 0.88049 0.88915
                          PC29    PC30    PC31    PC32    PC33    PC34    PC35
Standard deviation     0.75202 0.73547 0.70409 0.66661 0.65685 0.63397 0.62733
Proportion of Variance 0.00808 0.00773 0.00708 0.00635 0.00616 0.00574 0.00562
Cumulative Proportion  0.89723 0.90496 0.91204 0.91839 0.92455 0.93029 0.93591
                          PC36    PC37    PC38    PC39    PC40    PC41    PC42
Standard deviation     0.60563 0.58905 0.57919 0.54097 0.50603 0.50304 0.49794
Proportion of Variance 0.00524 0.00496 0.00479 0.00418 0.00366 0.00362 0.00354
Cumulative Proportion  0.94115 0.94611 0.95090 0.95508 0.95874 0.96236 0.96590
                          PC43    PC44   PC45    PC46    PC47    PC48    PC49
Standard deviation     0.47189 0.45320 0.4267 0.41580 0.40418 0.38997 0.36174
Proportion of Variance 0.00318 0.00293 0.0026 0.00247 0.00233 0.00217 0.00187
Cumulative Proportion  0.96908 0.97201 0.9746 0.97709 0.97942 0.98159 0.98346
                          PC50   PC51   PC52    PC53    PC54    PC55    PC56
Standard deviation     0.34997 0.3342 0.3132 0.30401 0.28847 0.28261 0.27847
Proportion of Variance 0.00175 0.0016 0.0014 0.00132 0.00119 0.00114 0.00111
Cumulative Proportion  0.98521 0.9868 0.9882 0.98953 0.99072 0.99186 0.99297
                          PC57    PC58    PC59    PC60    PC61    PC62   PC63
Standard deviation     0.27559 0.25568 0.23849 0.21748 0.21062 0.20665 0.1874
Proportion of Variance 0.00108 0.00093 0.00081 0.00068 0.00063 0.00061 0.0005
Cumulative Proportion  0.99405 0.99498 0.99580 0.99647 0.99711 0.99772 0.9982
                          PC64    PC65   PC66    PC67    PC68    PC69    PC70
Standard deviation     0.17247 0.15616 0.1456 0.13630 0.12118 0.10505 0.07161
Proportion of Variance 0.00042 0.00035 0.0003 0.00027 0.00021 0.00016 0.00007
Cumulative Proportion  0.99864 0.99899 0.9993 0.99956 0.99977 0.99993 1.00000
```

```r
## Compute amount of variability of components - sqrt of eigenvalues stored in sdev
sdev <- pca.vars$sdev^2
perc.expl <- sdev / sum(sdev)
## Calculate number of components needed to capture 80% of the variability
variability <- data.frame(variability = sort(perc.expl, decreasing=TRUE))
variability$cumulative <- cumsum(variability$variability)
num.comp.var.80 <- which.min(abs(variability$cumulative-0.8))
num.comp.var.70 <- which.min(abs(variability$cumulative-0.7))
num.comp.var.60 <- which.min(abs(variability$cumulative-0.6))
num.comp.var.1se <- which.min(abs(pca.vars$sdev-1.0))
```

```r
var.capture <- data.table("no. PCs",num.comp.var.80, num.comp.var.70,
                          num.comp.var.60, num.comp.var.1se)
colnames(var.capture) <- c("","80%", "70%", "60%", "1 std.error")
kable(var.capture, caption = "Number of principal components needed for explainability") |
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 25: Number of principal components needed for explainability

	80%	70%	60%	1 std.error
no. PCs	20	14	9	19

```r
## New data set of PC and factor variables
PCA_train_data.dt <- data.table(cancer_train.dt[,c("Event","Diam",
"LymphNodes","EstrogenReceptor","Grade","Age")],pca.vars$x[,1:19])
## Generate PCs for test set data
cancer_test.PCA.dt <- predict(pca.vars, newdata = cancer_test.dt[, ..gene.cols])[,1:19]
PCA_x_test_data.dt <- data.table(cancer_test.dt[,c("Event","Diam",
"LymphNodes","EstrogenReceptor","Grade","Age")],cancer_test.PCA.dt)
mod_matrix_PCA_test_data.dt <- model.matrix(Event ~ .,PCA_x_test_data.dt)
## Store the outcome and the covariates separately
y_PCA_train_data.dt <- PCA_train_data.dt$Event
x_PCA_train_data.dt <- PCA_train_data.dt[,!"Event"]
mod_matrix_PCA_train_data.dt <- model.matrix(Event ~ .,PCA_train_data.dt)
```

Of the 70 principal components for the training set data, it is seen below that 80% of the variability can be captured with 20 components, and 19 components can explain one standard deviation of the data. In this case, 19 components are selected to use in model construction. A new data set with the principal components for the gene expressions is built, with the categorical variables of `lymph nodes`, `diameter`, `age`, `estrogen receptor`, and `grade` also included.

```r
## Full PCA logistic regression
model.PCA <- glm(Event ~ ., data = PCA_train_data.dt, family="binomial")
summary(model.PCA)
```

```

Call:
glm(formula = Event ~ ., family = "binomial", data = PCA_train_data.dt)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.99996  -0.50053  -0.08473   0.24061   2.68872
```

```
Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)               6.19499    3.80100   1.630  0.10314
Diam>2cm                  0.64100    0.84183   0.761  0.44639
LymphNodes1-3            -2.43720    1.05507  -2.310  0.02089 *
EstrogenReceptorPositive  1.41520    2.11600   0.669  0.50362
GradePoorly diff          0.40293    0.99139   0.406  0.68443
GradeWell diff           -1.95463    1.40096  -1.395  0.16295
Age                      -0.16608    0.08594  -1.933  0.05328 .
PC1                       0.17661    0.17291   1.021  0.30708
PC2                      -0.29048    0.28140  -1.032  0.30194
PC3                      -0.17258    0.22805  -0.757  0.44918
PC4                      -0.57748    0.27226  -2.121  0.03392 *
PC5                      -0.36700    0.24733  -1.484  0.13785
PC6                      -0.35328    0.23946  -1.475  0.14013
PC7                      -0.62066    0.30890  -2.009  0.04451 *
PC8                      -0.01875    0.31669  -0.059  0.95280
PC9                      -0.93461    0.35956  -2.599  0.00934 **
PC10                     -0.09958    0.28780  -0.346  0.72933
PC11                      0.58316    0.34406   1.695  0.09009 .
PC12                     -0.20386    0.28317  -0.720  0.47157
PC13                      0.66832    0.42022   1.590  0.11175
PC14                     -0.65288    0.32152  -2.031  0.04230 *
PC15                     -0.49290    0.36018  -1.368  0.17117
PC16                      0.17362    0.36402   0.477  0.63339
PC17                     -1.31455    0.53103  -2.475  0.01331 *
PC18                      0.21621    0.37528   0.576  0.56454
PC19                      1.17737    0.41106   2.864  0.00418 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 129.849  on 101  degrees of freedom
Residual deviance:  61.747  on  76  degrees of freedom
AIC: 113.75

Number of Fisher Scoring iterations: 7
```

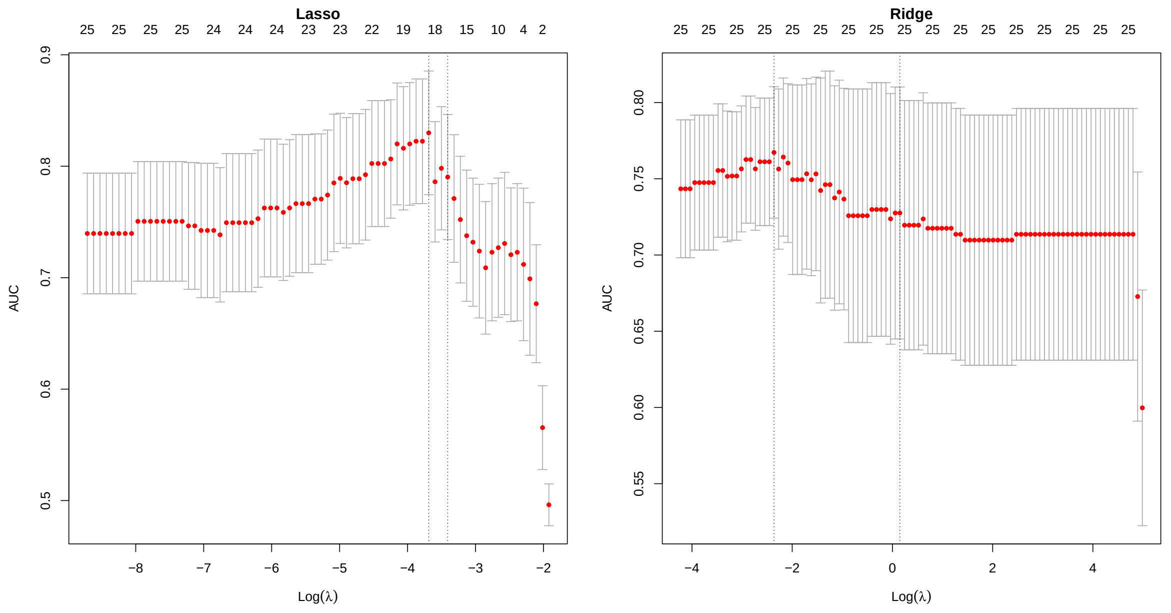
```r
## Backward stepwise selection
full.model <- glm(Event ~ ., data = PCA_train_data.dt, family = "binomial")
model.B.PCA <- stepAIC(full.model, direction="back", trace = FALSE)
summary(model.B.PCA)
```

```
Call:
glm(formula = Event ~ LymphNodes + Age + PC1 + PC4 + PC5 + PC7 +
    PC9 + PC11 + PC13 + PC14 + PC15 + PC17 + PC19, family = "binomial",
    data = PCA_train_data.dt)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6313  -0.5717  -0.1437   0.3462   2.4863

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)    4.84089    3.23465   1.497  0.13450
LymphNodes1-3 -2.07024    0.74206  -2.790  0.00527 **
Age           -0.10852    0.07162  -1.515  0.12973
PC1            0.24250    0.08989   2.698  0.00698 **
PC4           -0.44938    0.21705  -2.070  0.03842 *
PC5           -0.32707    0.19449  -1.682  0.09264 .
PC7           -0.36565    0.21787  -1.678  0.09328 .
PC9           -0.75779    0.28631  -2.647  0.00813 **
PC11           0.62812    0.33553   1.872  0.06121 .
PC13           0.63121    0.30631   2.061  0.03933 *
PC14          -0.67873    0.29223  -2.323  0.02020 *
PC15          -0.45298    0.31475  -1.439  0.15009
PC17          -0.78077    0.35117  -2.223  0.02619 *
PC19           1.15821    0.36805   3.147  0.00165 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 129.849  on 101  degrees of freedom
Residual deviance:  69.275  on  88  degrees of freedom
AIC: 97.275

Number of Fisher Scoring iterations: 6
```

```r
## Forward stepwise selection
null.model <- glm(Event ~ 1, data = PCA_train_data.dt, family = "binomial")
model.S.PCA <- stepAIC(null.model, scope = list(upper=full.model),
                        trace = FALSE, direction = "forward")
summary(model.S.PCA)
```

```
Call:
glm(formula = Event ~ LymphNodes + PC19 + PC1 + PC9 + PC14 +
    PC11 + PC4 + PC5 + PC17 + PC13 + PC15 + PC7 + Age, family = "binomial",
    data = PCA_train_data.dt)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6313  -0.5717  -0.1437   0.3462   2.4863

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)    4.84089    3.23465   1.497  0.13450
LymphNodes1-3 -2.07024    0.74206  -2.790  0.00527 **
PC19           1.15821    0.36805   3.147  0.00165 **
PC1            0.24250    0.08989   2.698  0.00698 **
PC9           -0.75779    0.28631  -2.647  0.00813 **
PC14          -0.67873    0.29223  -2.323  0.02020 *
PC11           0.62812    0.33553   1.872  0.06121 .
PC4           -0.44938    0.21705  -2.070  0.03842 *
PC5           -0.32707    0.19449  -1.682  0.09264 .
PC17          -0.78077    0.35117  -2.223  0.02619 *
PC13           0.63121    0.30631   2.061  0.03933 *
PC15          -0.45298    0.31475  -1.439  0.15009
PC7           -0.36565    0.21787  -1.678  0.09328 .
Age           -0.10852    0.07162  -1.515  0.12973
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 129.849  on 101  degrees of freedom
Residual deviance:  69.275  on  88  degrees of freedom
AIC: 97.275

Number of Fisher Scoring iterations: 6
```

```r
## Fit lasso and ridge models on PCA data
fit.cv.lasso.PCA <- cv.glmnet(mod_matrix_PCA_train_data.dt, y_PCA_train_data.dt,
family = "binomial", type.measure = "auc", alpha = 1)
fit.cv.ridge.PCA <- cv.glmnet(mod_matrix_PCA_train_data.dt, y_PCA_train_data.dt,
family = "binomial", type.measure = "auc", alpha = 0)
## Plot the cross-validation curves
par(mfrow=c(1,2), mar=c(4,4,5,2))
plot(fit.cv.lasso.PCA, main="Lasso")
plot(fit.cv.ridge.PCA, main="Ridge")
```



```r
## Find indices of lambdas that maximizes AUC
idx_lasso.PCA.min <- fit.cv.lasso.PCA$index["min",]
idx_ridge.PCA.min <- fit.cv.ridge.PCA$index["min",]

## Reporting model size of each regression
modelsize <- data.table("Model Size",
                        fit.cv.lasso.PCA$nzero[idx_lasso.PCA.min],
                        fit.cv.ridge.PCA$nzero[idx_ridge.PCA.min])
colnames(modelsize) <- c("","Lasso", "Ridge")
kable(modelsize,
  caption = "Model Size of Lasso and Ridge Regression") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 26: Model Size of Lasso and Ridge Regression

	Lasso	Ridge
Model Size	19	25

Running a full logistic regression shows that most variables are not significant. This model may not be optimal, and covariate selection can be aided by backwards elimination and forward selection. To select which components to keep for the best fit, backwards elimination and forward selection are performed. Although the two models were created through two different methods, both selection processes have produced the exact same model. Therefore analysis can continue with a model titled "Logistic regression PCA" to represent the model selected by both backwards elimination and forward selection. The PCA data set is also used to fit lasso and ridge regression models. With the dimensionality reduction, it is still found that ridge regression is using all possible covariates, where lasso produces a maximum AUC fit with 19 covariates.

Training Set Model Accuracy

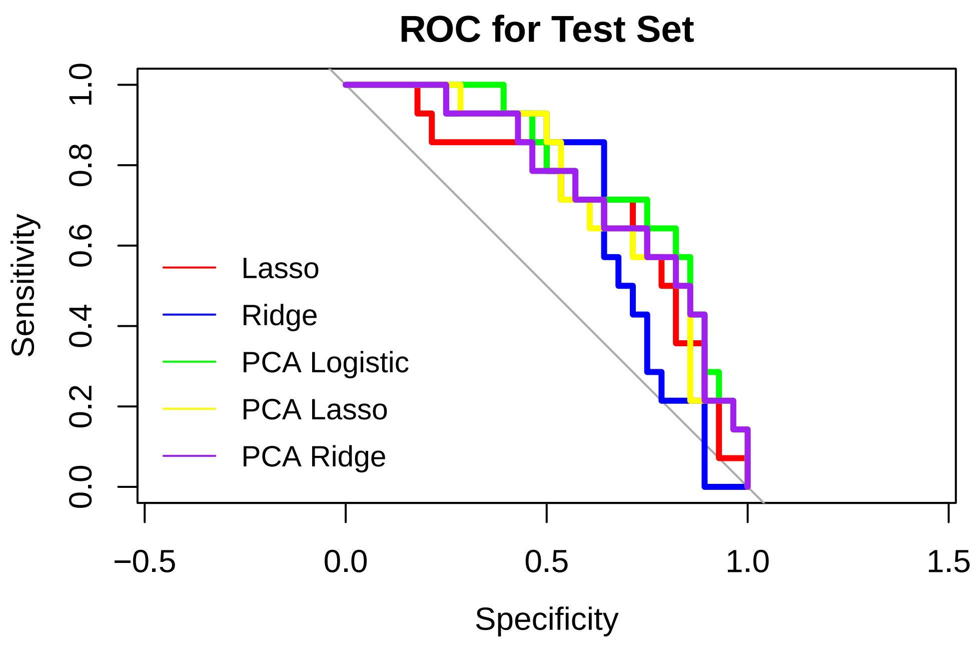
```r
## Data frame of actual data observations for the outcome in train set,
## and predicted outcome from the model, compare predictive ability
## Lasso train prediction
suppressMessages(invisible({
  pred_model.lasso <- data.frame(obs = cancer_train.dt$Event,
                          pred = predict(fit.cv.lasso, s = fit.cv.lasso$lambda.min,
                          newx = x_cancer_train.dt, type = "response"))
  auc_model.lasso <- roc(obs ~ s1, data = pred_model.lasso)$auc
  ## Ridge train prediction
  pred_model.ridge <- data.frame(obs = cancer_train.dt$Event,
                          pred = predict(fit.cv.ridge, s = fit.cv.ridge$lambda.min,
                          newx = x_cancer_train.dt, type = "response"))
  auc_model.ridge <- roc(obs ~ s1, data = pred_model.ridge)$auc
  ## Logistic regression PCA train prediction
  pred_model.B.PCA <- data.frame(obs = cancer_train.dt$Event,
                          pred = predict(model.B.PCA, newdata = x_PCA_train_data.dt,
                          type = "response"))
  auc_model.B.PCA <- roc(obs ~ pred, data = pred_model.B.PCA)$auc
  ## Lasso PCA train prediction
  pred_model.lasso.PCA <- data.frame(obs = cancer_train.dt$Event,
                          pred = predict(fit.cv.lasso.PCA, s = fit.cv.lasso.PCA$lambda.min,
                          newx = mod_matrix_PCA_train_data.dt, type = "response"))
  auc_model.lasso.PCA <- roc(obs ~ s1, data = pred_model.lasso.PCA)$auc
  ## Ridge PCA train prediction
  pred_model.ridge.PCA <- data.frame(obs = cancer_train.dt$Event,
                       pred = predict(fit.cv.ridge.PCA, s = fit.cv.ridge.PCA$lambda.min,
                       newx = mod_matrix_PCA_train_data.dt, type = "response"))
  auc_model.ridge.PCA <- roc(obs ~ s1, data = pred_model.ridge.PCA)$auc
}))
```

To evaluate the accuracy of each model on the training data, the model is used to predict the training set outcomes, and compared to the actual outcomes. The training AUC values are calculated, which represents how well each model predicts true positives and true negatives, and will be presented in an overall comparison in the upcoming section.

Final Predictive Accuracy Comparison

```r
suppressMessages(invisible({
  ## Lasso test prediction
  test_pred_model.lasso <- data.frame(obs = cancer_test.dt$Event,
                  pred = predict(fit.cv.lasso, s = fit.cv.lasso$lambda.min,
                  newx = x_cancer_test.dt, type = "response"))
  test_auc_model.lasso <- roc(obs ~ s1, data = test_pred_model.lasso)$auc
  ## Ridge test prediction
  test_pred_model.ridge <- data.frame(obs = cancer_test.dt$Event,
                  pred = predict(fit.cv.ridge, s = fit.cv.ridge$lambda.min,
                  newx = x_cancer_test.dt, type = "response"))
  test_auc_model.ridge <- roc(obs ~ s1, data = test_pred_model.ridge)$auc
  ## Logistic regression PCA test prediction
  test_pred_model.B.PCA <- data.frame(obs = cancer_test.dt$Event,
                  pred = predict(model.B.PCA, newdata = PCA_x_test_data.dt,
                  type = "response"))
  test_auc_model.B.PCA <- roc(obs ~ pred, data = test_pred_model.B.PCA)$auc
  ## Lasso PCA test prediction
  test_pred_model.lasso.PCA <- data.frame(obs = cancer_test.dt$Event,
                  pred = predict(fit.cv.lasso.PCA, s = fit.cv.lasso.PCA$lambda.min,
                  newx = mod_matrix_PCA_test_data.dt, type = "response"))
  test_auc_model.lasso.PCA <- roc(obs ~ s1, data = test_pred_model.lasso.PCA)$auc
  ## Ridge PCA test prediction
  test_pred_model.ridge.PCA <- data.frame(obs = cancer_test.dt$Event,
                  pred = predict(fit.cv.ridge.PCA, s = fit.cv.ridge.PCA$lambda.min,
                  newx = mod_matrix_PCA_test_data.dt, type = "response"))
  test_auc_model.ridge.PCA <- roc(obs ~ s1, data = test_pred_model.ridge.PCA)$auc
  }))
```

```
## Plot ROC curve for all models
par(mfrow=c(1,1))
suppressMessages(invisible({
  roc(cancer_test.dt$Event, test_pred_model.lasso$s1, plot = TRUE, lwd = 3,
      xlim = c(0,1), col="red", main="ROC for Test Set")
  plot.roc(cancer_test.dt$Event, test_pred_model.ridge$s1, add = TRUE,
           lwd = 3, col="blue", xlim = c(0,1))
  plot.roc(cancer_test.dt$Event, test_pred_model.B.PCA$pred, add = TRUE,
           lwd = 3, col="green", xlim = c(0,1))
  plot.roc(cancer_test.dt$Event, test_pred_model.lasso.PCA$s1, add = TRUE,
           lwd = 3, col="yellow", xlim = c(0,1))
  plot.roc(cancer_test.dt$Event, test_pred_model.ridge.PCA$s1, add = TRUE,
           lwd = 3, col="purple", xlim = c(0,1))
}))
legend("bottomleft",
        legend = c("Lasso", "Ridge", "PCA Logistic", "PCA Lasso","PCA Ridge"),
        col = c("red", "blue", "green", "yellow", "purple"),
        cex = .9, lty = 1, bty ="n")
```



```r
## Create table of AUC values
AUC_table <- data.table(
c("Lasso regression","Ridge regression","PCA logistic regression",
"PCA lasso regression","PCA ridge regression"),
round(c(auc_model.lasso, auc_model.ridge, auc_model.B.PCA,
auc_model.lasso.PCA, auc_model.ridge.PCA),3),
round(c(test_auc_model.lasso, test_auc_model.ridge, test_auc_model.B.PCA,
test_auc_model.lasso.PCA,test_auc_model.ridge.PCA),3))
colnames(AUC_table) <- c("Model type", "Training AUC Value", "Test AUC Value")
kable(AUC_table, caption = "AUC values for Training and Testing Sets") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 27: AUC values for Training and Testing Sets

Model type	Training AUC Value	Test AUC Value
Lasso regression	0.927	0.712
Ridge regression	0.829	0.691
PCA logistic regression	0.908	0.778
PCA lasso regression	0.915	0.742
PCA ridge regression	0.923	0.745

To compare the predictive accuracy of each model, the models are used to predict the outcome of the test set, which was not used to construct the models. These predictions are then compared against the actual test set values, and the AUC values compared in the table below. Both training and test AUC are reported, as the relationship between how the model fits both sets of data are important. Test set accuracy is most important, and high training set AUC values with low test set AUC values indicates overfitting may have occurrred which is not desirable in a model.

It appears that the PCA logistic regression model has the best predictive ability, with a test AUC value of 0.778 indicating that with this model there is a 77.8% chance that it will correctly distinguish between "event" and "no event" based on the data.

```r
## Calculate confusion matrix objects for all models
sens_model.lasso <- confusionMatrix(as.factor(test_pred_model.lasso$obs),
                              as.factor(round(test_pred_model.lasso$s1,0)))
sens_model.B.PCA <- confusionMatrix(as.factor(test_pred_model.B.PCA$obs),
                              as.factor(round(test_pred_model.B.PCA$pred,0)))
sens_model.lasso.PCA <- confusionMatrix(as.factor(test_pred_model.lasso.PCA$obs),
                              as.factor(round(test_pred_model.lasso.PCA$s1,0)))
sens_model.ridge.PCA <- confusionMatrix(as.factor(test_pred_model.ridge.PCA$obs),
                              as.factor(round(test_pred_model.ridge.PCA$s1,0)))
```

```r
sens_table <- data.table(c("Lasso regression","PCA logistic regression",
                           "PCA lasso regression","PCA ridge regression"),
                   round(c(sens_model.lasso[[4]][1], sens_model.B.PCA[[4]][1],
                          sens_model.lasso.PCA[[4]][1], sens_model.ridge.PCA[[4]][1]),3),
                   round(c(sens_model.lasso[[4]][2], sens_model.B.PCA[[4]][2],
                          sens_model.lasso.PCA[[4]][2], sens_model.ridge.PCA[[4]][2]),3))
colnames(sens_table) <- c("Model type", "Sensitivity","Specificity")
kable(sens_table, caption = "Confusion Matrix") |>
kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 28: Confusion Matrix

Model type	Sensitivity	Specificity
Lasso regression	0.742	0.545
PCA logistic regression	0.815	0.600
PCA lasso regression	0.750	0.600
PCA ridge regression	0.774	0.636

As mentioned above, sensitivity is very important in this context, due to the damage that can be done if a true positive is not identified correctly. Sensitivity and specificity of the four best performing models is compared below. Ridge regression is excluded due to it being a very complex model and lacking performance compared to the others. With the highest sensitivity, the PCA logistic regression model is performing the best of all the models for predictive power for identifying patients who are associated with having a post-surgery event. Therefore this model would be recommended for use to assist in providing prognoses to patients.