Supplementary R Code

# Load packages

library(janitor)  
library(dplyr)  
library(caret)

# Create dataset

#limit to stages 1 & 2, having surgery but no chemotherapy  
  
metaclinicalfinal <- metaclinical %>% filter((metaclinical$tumor\_stage == 1 | metaclinical$tumor\_stage == 2) & (metaclinical$type\_of\_breast\_surgery == "BREAST CONSERVING" | metaclinical$type\_of\_breast\_surgery == "MASTECTOMY") & (metaclinical$chemotherapy == "NO"))  
  
#Merge clinical with genomic data  
  
subset <- metaclinicalfinal %>% select(patient\_id, relapse\_free\_status2, radio\_therapy) %>% filter(radio\_therapy == "YES")  
metamerge <- merge(metagene, subset, by = "patient\_id")   
metamerge <- subset(metamerge, select = -c(patient\_id, radio\_therapy))

# Wilcoxon Signed Rank test

pValuesall.wilcox <- apply(metamerge[,1:24369], 2, function(x) wilcox.test(x[1:292],x[293:461])$p.value) #rows 1-292 and 293-461 split patients with and without a recurrence  
pValuesall.wilcox <- as.data.frame(pValuesall.wilcox)  
reduce <- pValuesall.wilcox %>% filter(pValues < 0.05) #filter by p<0.05

# Cox Proportional Hazards

cox\_models <- lapply(metamerge[,1:24369], function(x){coxph(Surv(relapse\_free\_status\_months, relapse\_free\_status3) ~ x,data=metamerge)}) #variable relapse\_free\_status\_months must be added back to the df  
  
pValues <- sapply(1:length(cox\_models), function(x) {summary(cox\_models[[x]])$coefficients[5]})  
HR <- sapply(1:length(cox\_models), function(x) {summary(cox\_models[[x]])$coefficients[2]})  
CI <- sapply(1:length(cox\_models), function(x) {exp(confint(cox\_models[[x]]))})  
genes <- colnames(metagene[,3:24370])  
  
cox\_allgene\_pvals <- rbind(genes,HR,pValues,CI)  
cox\_allgene\_pvals <- as.data.frame(t(cox\_allgene\_pvals))  
reduce <- cox\_allgene\_pvals %>% filter(pValues < 0.05) #filter by p<0.05

# Create random sets of insignifant genes

set.seed(100)  
insigsubset <- df[,sample(ncol(df), 1595)]

# Run Support Vector Machine

set.seed(123)  
cox.df.all\_split <- createDataPartition(cox.df.all$relapse\_free\_status2, p = .8,   
 list = FALSE,   
 times = 1)  
train <- cox.df.all[ cox.df.all\_split,]  
test <- cox.df.all[-cox.df.all\_split,]  
  
svmall <- train(relapse\_free\_status2 ~ ., data = train,   
 method = "svmLinear",   
 trControl = trainControl(## 10-fold Cross validation  
 method = "cv",  
 number = 10 ))  
svmall  
  
test$relapse\_free\_status2 <- factor(test$relapse\_free\_status2, levels = c("DiseaseFree", "Recurred/Progressed"))  
test\_pred <- predict(svmall, newdata = test)  
confusionMatrix(test\_pred, test$relapse\_free\_status2)

# Tune cost hyperparameter for SVM

set.seed(123)  
  
df2\_split <- createDataPartition(df2$relapse\_free\_status2, p = .8,   
 list = FALSE,   
 times = 1)  
train <- df2[ df2\_split,]  
test <- df2[-df2\_split,]  
  
svmall <- train(relapse\_free\_status2 ~ ., data = train,   
 method = "svmLinear",   
 trControl = trainControl(method = "repeatedcv", number = 10, repeats = 3),  
 tuneGrid = expand.grid(C = seq(0, 2, length = 20)))   
   
svmall  
plot(svmall)  
confusionmatrix(svmall, positive = "Recurred/Progressed" )

# Run Neural Networks

set.seed(123)  
df2\_split <- createDataPartition(df2$relapse\_free\_status2, p = .8,   
 list = FALSE,   
 times = 1)  
train <- df2[ df2\_split,]  
test <- df2[-df2\_split,]  
  
nnetall <- train(relapse\_free\_status2 ~ ., data = train,   
 method = "nnet",   
 trControl = trainControl(## 10-fold CV  
 method = "cv",  
 number = 10),  
 ## repeated ten times  
 MaxNWts=5000)  
  
nnetall  
  
test$relapse\_free\_status2 <- factor(test$relapse\_free\_status2, levels = c("DiseaseFree", "Recurred/Progressed"))  
test\_pred <- predict(nnetall, newdata = test)  
confusionMatrix(test\_pred, test$relapse\_free\_status2)

# Tune decay for Neural Networks

set.seed(123)  
df2\_split <- createDataPartition(df2$relapse\_free\_status2, p = .8,   
 list = FALSE,   
 times = 1)  
train <- df2[ df2\_split,]  
test <- df2[-df2\_split,]  
  
NN <- train(relapse\_free\_status2 ~ ., data = train,   
 method = "nnet",   
 trControl = trainControl(method = "cv", number = 10),  
 tuneGrid = expand.grid(size = 3, decay = seq(0.1, 2.5, length = 10)) ,   
 MaxNWts=5000)  
   
NN  
  
test\_pred <- predict(NN, newdata = test)  
confusionMatrix(test\_pred, test$relapse\_free\_status2)

# Test model on RT-untreated patients

dfneg\_pred <- predict(nnetall, newdata = dfneg)   
confusionMatrix(dfneg\_pred, dfneg$relapse\_free\_status2)

# Recursive Feature Elimination

fs <- df2[,1:1596] #subset set of features  
  
rfe\_control = rfeControl(   
 method="cv",  
 number=10)  
  
fit\_control = trainControl(classProbs=F,  
 search="random")  
  
relapse <- as.factor(ifelse(df2$relapse\_free\_status2 == "DiseaseFree", "No", "Yes")) #code outcome variable   
   
rfe\_fit = rfe(fs, relapse,  
 sizes = c(10,20,30,40,50,60,70,80,90,100,250,500,750,1000),  
 rfeControl = rfe\_control,  
 method="svmLinear",  
 trControl=fit\_control)  
  
  
trellis.par.set(caretTheme())  
plot(rfe\_fit, type = c("g", "o")

# Build final Support Vector Machine model with 1596 genes, cost = 1

set.seed(123)  
df2\_split <- createDataPartition(df2$relapse\_free\_status2, p = .8,   
 list = FALSE,   
 times = 1)  
train <- df2[ df2\_split,]  
test <- df2[-df2\_split,]  
  
SVM1596 <- train(relapse\_free\_status2 ~ ., data = train,   
 method = "svmLinear",   
 trControl = trainControl(method = "repeatedcv", number = 10, repeats = 3, summaryFunction=twoClassSummary, classProbs=T, savePredictions = T), #for plotting of AUROC curves  
 tuneGrid = expand.grid(C=1))  
   
   
SVM1596  
  
test$relapse\_free\_status2 <- factor(test$relapse\_free\_status2, levels = c("Recurred.Progressed", "DiseaseFree"))  
test\_pred <- predict(SVM1596, newdata = test)  
confusionMatrix(test\_pred, test$relapse\_free\_status2)

# Comparison of all models

mods <- resamples(list(SVM977 = SVM977, SVM1596 = SVM1596, NN977 = NN977, NN1596 = NN1596))  
summary(mods)  
bwplot(mods) #box plots of models  
diffs <- diff(mods) # difference in model predictions  
summary(diffs) # summarize p-values for pair-wise comparisons

# Gene Set Enrichment analysis

if (!requireNamespace("BiocManager", quietly=TRUE))  
+ install.packages("BiocManager")  
BiocManager::install("topGO")  
  
if (!requireNamespace("BiocManager", quietly=TRUE))  
+ install.packages("BiocManager")  
BiocManager::install(c("KEGGREST", "org.Hs.eg.db", "Rgraphviz"))  
  
library(topGO)  
library(KEGGREST)  
library(org.Hs.eg.db)  
library(Rgraphviz)  
  
cox\_gene\_pvals <- read.csv("cox\_gene\_pvals.csv", header=T, row.names=1)  
reduce <- cox\_gene\_pvals %>% filter(pValues < 0.05)   
geneList <- reduce$pValues  
names(geneList) <- reduce$genes  
  
# Mapping all 977 genes to biological process   
GOdata <- new('topGOdata',  
 ontology = "BP", #character string specifying the ontology of interest (BP, MF or CC)  
 allGenes = geneList, #named vector of type numeric or factor. The names attribute contains the genes identifiers. The  
#genes listed in this object define the gene universe.  
 geneSelectionFun = function(x)x,   
 nodeSize = 10, #used to prune the GO hierarchy from the terms which have less than 10 annotated genes   
 annot = annFUN.org, mapping = "org.Hs.eg.db", ID = "SYMBOL") #function which maps genes identifiers to GO terms.  
  
resultKS <- runTest(GOdata, algorithm = "weight01", statistic = "ks") # Kolmogorov-Smirnov testing  
tab2 <- GenTable(GOdata, raw.p.value = resultKS, topNodes = length(resultKS@score), numChar = 120)

# Overlap with radiogenes

if (!requireNamespace("BiocManager", quietly = TRUE))  
 install.packages("BiocManager")  
  
BiocManager::install("org.Hs.eg.db")  
  
library(org.Hs.eg.db)  
keytypes(org.Hs.eg.db)  
columns(org.Hs.eg.db)  
cols <- c( "ENTREZID", "ALIAS", "GENENAME" ,"SYMBOL")  
  
#Search for aliases of radiogenes  
radiogenes = litradiogenes$gene #list of radiogenes   
search1 <- AnnotationDbi::select(org.Hs.eg.db, keys=radiogenes, columns=cols, keytype="SYMBOL")   
  
#search for aliases of geneset  
allgenes = geneset$gene #list of genes   
search2 <- AnnotationDbi::select(org.Hs.eg.db, keys=allgenes, columns=cols, keytype="SYMBOL")

#find overlap between aliases  
intersect(search1$ALIAS, search2$ALIAS)  
length(intersect(search1$ALIAS, search3$ALIAS)) #46

# Timing model performance

ptm <- proc.time()   
test\_pred <- predict(SVM1596, newdata = test)  
confusionMatrix(test\_pred, test$relapse\_free\_status2)  
proc.time() - ptm