Sepsis Detection Final Project

Elena Wang, Songyun Zhang, Zhen Liu

11/29/2022

Upload and Combine DatasetA and B, and Convert to Correct Data Type

```
trd <- read.csv("/Users/xiaoy/OneDrive/Desktop/BIOSTA707_FinalProject/dataset/training_setA/trai
ningA_final.csv", header=TRUE)
ted <- read.csv("/Users/xiaoy/OneDrive/Desktop/BIOSTA707_FinalProject/dataset/training_setB/test
ingB_final.csv", header=TRUE)
dat <- rbind(trd,ted)
dat$Gender = as.factor(dat$Gender)
dat$SepsisLabel = as.factor(dat$SepsisLabel)
trd$Gender = as.factor(trd$Gender)
trd$SepsisLabel = as.factor(trd$SepsisLabel)
ted$Gender = as.factor(ted$Gender)
ted$SepsisLabel = as.factor(ted$SepsisLabel)
```

Transform the imbalanced dataset

reduce the perpentage of 0 in both

```
#Show the imbalance
table(trd$SepsisLabel)
```

```
##
##     0     1
## 18546  1790
```

```
table(ted$SepsisLabel)
```

```
##
##     0     1
## 18858  1142
```

```
nrow(trd %>% filter(SepsisLabel==0))/nrow(trd)
```

```
## [1] 0.9119788
```

```
nrow(ted %>% filter(SepsisLabel==0))/nrow(ted)
```

```
## [1] 0.9429
```

```
# imbalance train
set.seed(456)
ind.trd <- sample(seq(nrow(trd %>% filter(SepsisLabel==0))),size = round(nrow(trd %>% filter(Sep
sisLabel==0))*0.1),replace = FALSE)
trd.new <- rbind(trd[ind.trd,],trd %>%
                   filter(SepsisLabel==1))
table(trd.new$SepsisLabel)
```

```
##
##    0    1
## 1670 1975
```

```
nrow(trd.new %>% filter(SepsisLabel==0))/nrow(trd.new)
```

```
## [1] 0.4581619
```

```
#Test dataset
set.seed(456)
ind.ted <-sample(seq(nrow(ted %>%
filter(SepsisLabel==0))),size = round(nrow(ted %>% filter(SepsisLabel==0))*0.1),replace =FALSE)
ted.new <- rbind(ted[ind.ted,],ted%>%
                   filter(SepsisLabel == 1))
table(ted.new$SepsisLabel)
```

```
##
##    0    1
## 1764 1264
```

```
nrow(ted.new %>% filter(SepsisLabel==0))/nrow(ted.new)
```

```
## [1] 0.5825627
```

```
df <- rbind(trd.new,ted.new)
```

EDA

Distribution Checking and Standardization

```
df_num = select_if(df, is.numeric)
# Standardization
df_std = cbind(df[,c("SubjectID","Gender","SepsisLabel")], scale(df_num,center= TRUE, scale = TR
UE))
df_num = df_std[,c(-1,-2,-3)]
```

```
#par(mfrow=c(1,2))
##Plot the distribution for the original dataset
#df %>%
#  keep(is.numeric) %>%                     # Keep only #numeric columns
#  gather() %>%                             # Convert to #key-value pairs
#  ggplot(aes(value)) +                     # Plot the #values
#  facet_wrap(~key, scales = "free") +   # In separate #panels
#  geom_histogram(bins=60,  color="skyblue", #fill="skyblue",alpha = 0.5)   +
#  geom_density(alpha=0.5) +
#  ggtitle("Distribution of Covariates for Original #Dataset")

#Plot the distribution for the standardized dataset
df_std %>%
  keep(is.numeric) %>%                     # Keep only numeric columns
  gather() %>%                             # Convert to key-value pairs
  ggplot(aes(value)) +                     # Plot the values
  facet_wrap(~key, scales = "free") +   # In separate panels
  geom_histogram(bins=60, color="skyblue", fill="skyblue",alpha = 0.5)   +
  geom_density(alpha=0.5)
```

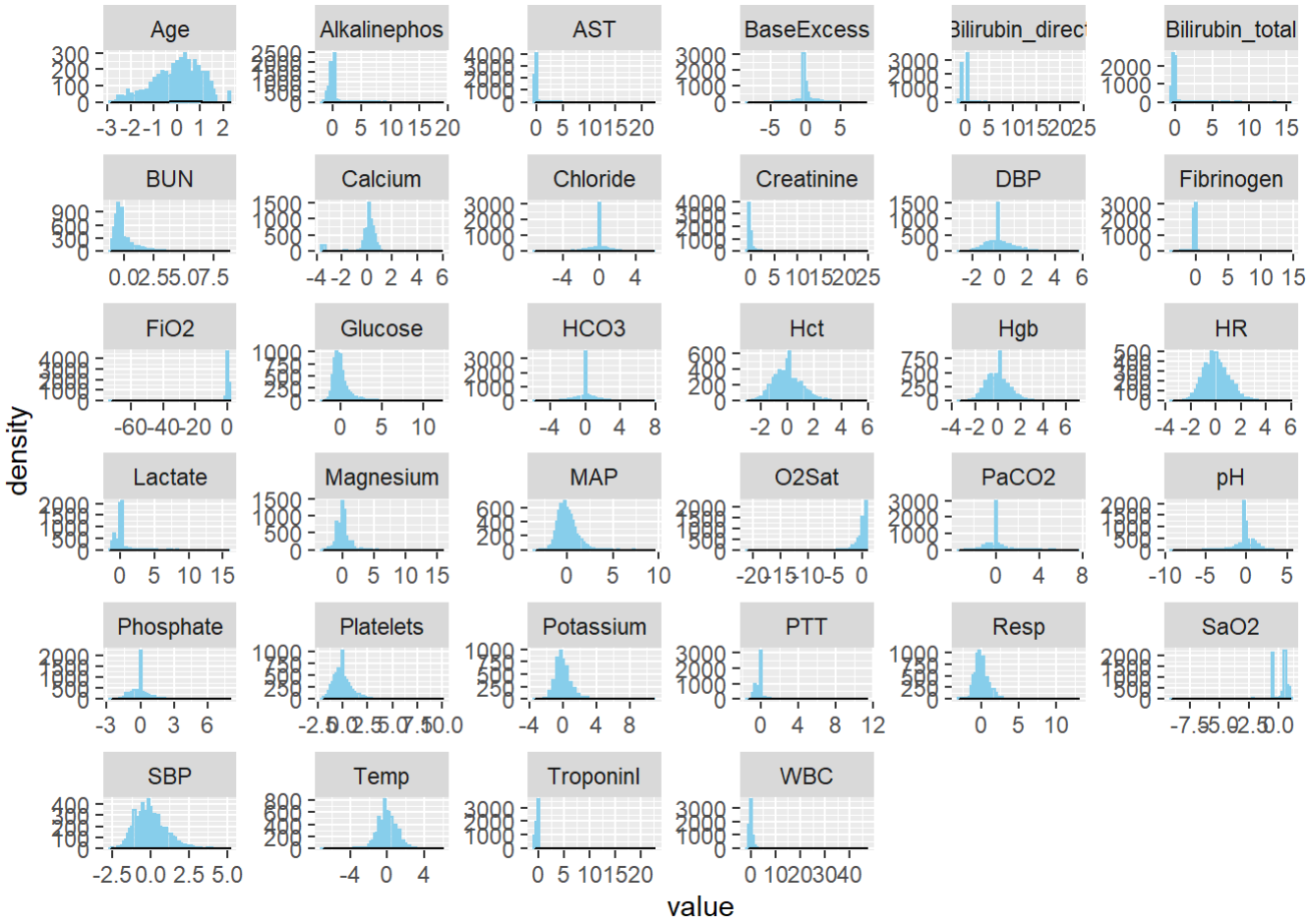

Figure1: Distribution of Covariates After Standardized

Gender vs Sepsis

```
library(scales)
# create segmented bar chart
# adding labels to each segment
plotdata <- df_std %>%
  group_by(Gender, SepsisLabel) %>%
  summarize(n = n()) %>%
  mutate(pct = n/sum(n),
         lbl = scales::percent(pct)) %>%
   mutate(Gender = if(0 %in% Gender) "Female" else "Male")

ggplot(plotdata,
       aes(x = factor(Gender,
                      levels = c("Male", "Female")),
           y = pct,
           fill = factor(SepsisLabel,
                         levels = c("1", "0"),
                         labels = c("sepsis", "nonsepsis")))) +
  geom_bar(stat = "identity",
           position = "fill") +
  scale_y_continuous(breaks = seq(0, 1, .2),
                     label = percent) +
  geom_text(aes(label = lbl),
            size = 3,
            position = position_stack(vjust = 0.5)) +
  scale_fill_brewer(palette = "Set2") +
  labs(y = "Percent",
       fill = "Sepsis",
       x = "Gender") +
  theme_minimal()
```

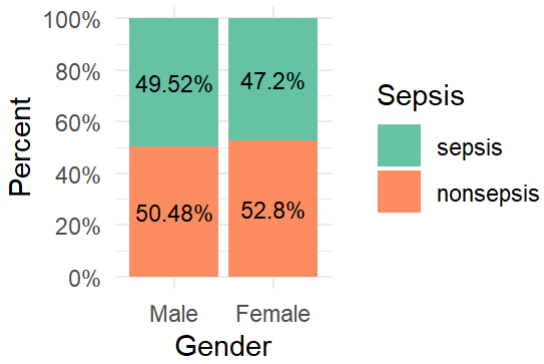

Figure2: Gender vs. Sepsis

Correlation Plot

```
#Correlation plot
#M = cor(df_num)
#corrplot(M, method = "color") # colorful number
corrplot.mixed(cor(df_num), lower = "circle",
               upper = "number",
               tl.col = "black", tl.cex=0.5, number.cex=0.5,number.digits=1)
```

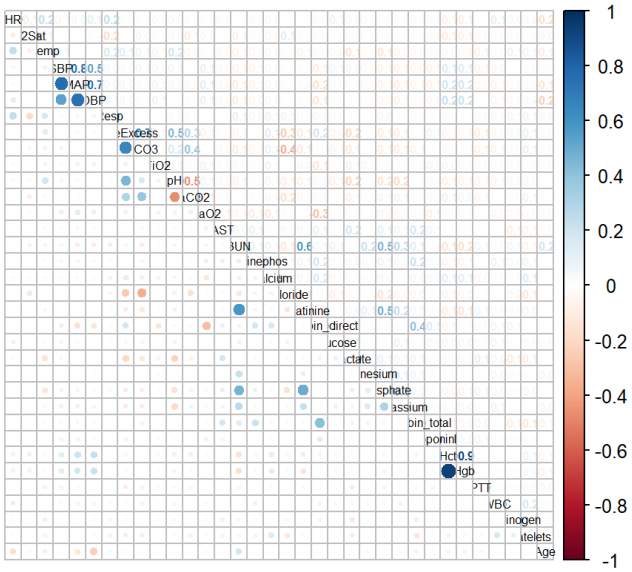

Figure3: Correlation Plot

PCA

```
# PCA
pca_sep <- prcomp(df_num)
scores <- data.frame(df_std, pca_sep$x[,1:3])
pc1.2 <- qplot(x=PC1, y=PC2, data=scores, color = as.factor(SepsisLabel))+
    labs(colour = 'Spesis Label')+ xlab("PC1(8.9%)") + ylab("PC2(7.5%)")
pc1.2
```

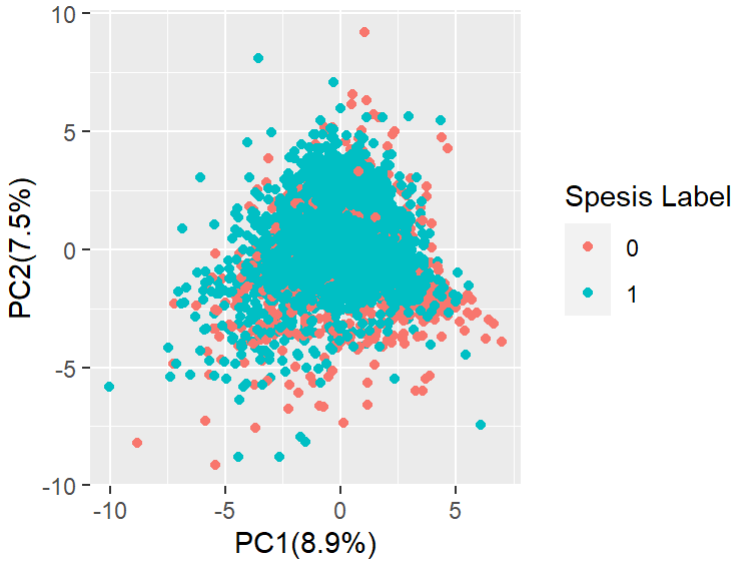

Figure4: PC1 vs PC2

Kmeans Clustering

```
# Kmeans clustering
km.out=kmeans(df_num,2,nstart=20)
table(df_std$SepsisLabel,km.out$cluster)
```

```
##
##        1    2
##   0 1675 1759
##   1 1125 2114
```

```
fviz_cluster(km.out, data = df_num,geom = "point",
ellipse.type = "euclid",ggtheme = theme_minimal(),
palette = "Set2")
```

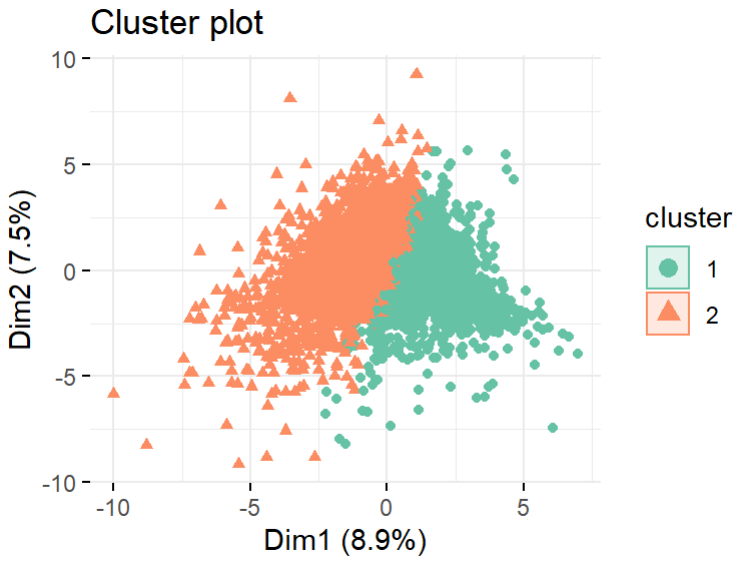

Figure5: K-means clustering

```
# Hierarchical clustering
#hc.complete=hclust(dist(df_num), method="complete")
#plot(hc.complete)
```

Modelling

Cross Validated Logistic Regression with Lasso Penalty to Select Important Variables

```
# Separate training and testing
trd = df_std[1:nrow(trd.new),]
ted = df_std %>% anti_join(trd)
# create dummy variable for gender
trd = trd %>% mutate(Gender = case_when(Gender == "1"~1,Gender == "0"~0))
ted = ted %>% mutate(Gender = case_when(Gender == "1"~1,Gender == "0"~0))

x_train <- as.matrix(trd%>% dplyr::select(-SubjectID,-SepsisLabel))
x_test <- as.matrix(ted%>% dplyr::select(-SubjectID,-SepsisLabel))
y_train <- as.matrix(trd %>% dplyr::select(SepsisLabel))
y_test <- as.matrix(ted %>% dplyr::select(SepsisLabel))
```

```
## conduct variable selection using cross validated lasso
set.seed(2022)
cv.lasso <- cv.glmnet(x_train, y_train, family="binomial", alpha = 1)
best_lambda <- cv.lasso$lambda.min # 0.003790231 is the best lambda that minimize MSE
lasso_result <- coef(cv.lasso)
lasso.choice <- rownames(lasso_result)[which(lasso_result !=0)]
lasso.choice
```

```
##  [1] "(Intercept)" "O2Sat"       "Temp"        "MAP"         "Resp"
##  [6] "FiO2"        "pH"          "BUN"         "Lactate"     "Magnesium"
## [11] "Phosphate"   "Hgb"         "PTT"         "WBC"         "Fibrinogen"
## [16] "Platelets"
```

```
pred_train = round(predict(cv.lasso, s = cv.lasso$lambda.min, newx = x_train,type = "response"))
pred_test = round(predict(cv.lasso, s = cv.lasso$lambda.min, newx = x_test,type = "response"))
# Model accuracy
accurary_lassotr = mean(pred_train == y_train)
accurary_lassote = mean(pred_test == y_test) # 71.17% accuracy
# ErrorRate
error_lassotr = mean(pred_train != y_train)
error_lassote = mean(pred_test != y_test) # 28.83% error Rate
```

```
### After the variable selection, we got our new train and test dataset.
select_cols = lasso.choice[-1]
x_train.dt <- setDF(trd%>% dplyr::select(-SubjectID,-SepsisLabel))
x_test.dt <-  setDF(ted%>% dplyr::select(-SubjectID,-SepsisLabel))
setDT(x_train.dt)
setDT(x_test.dt)
lasso_trainx <- x_train.dt[ , ..select_cols]
lasso_testx <- x_test.dt[ , ..select_cols]
```

Logisitc regression, LDA & QDA

```
# logistic regression
logistic.tr <- glm(as.factor(y_train) ~.,family=binomial(link='logit'), data= lasso_trainx)
# Make predictions
pred_train = round(logistic.tr%>%predict(as.data.frame(lasso_trainx), type = "response"))
pred_test = round(logistic.tr%>%predict(as.data.frame(lasso_testx), type = "response"))
# Model accuracy
accuracy_logtr = mean(pred_train == y_train)
accuracy_logte = mean(pred_test == y_test) # 72.06% accuracy
# ErrorRate
error_logtr = mean(pred_train != y_train)
error_logte = mean(pred_test != y_test) # 27.94% error Rate
```

```
# LDA
# Fit the model
lda <- lda(as.factor(y_train)~., data = lasso_trainx)
# Make predictions
pred_train = lda%>%predict(lasso_trainx)
pred_test = lda%>%predict(lasso_testx)
# Model accuracy
accuracy_ldatr = mean(pred_train$class == y_train)
accuracy_ldate = mean(pred_test$class == y_test) # 71.86% accuracy
# ErrorRate
error_ldatr = mean(pred_train$class != y_train)
error_ldate = mean(pred_test$class != y_test) # 28.14% error Rate
```

```
# QDA
# Fit the model
qda <- qda(as.factor(y_train)~., data = lasso_trainx)
# Make predictions
pred_train = qda%>%predict(lasso_trainx)
pred_test = qda%>%predict(lasso_testx)
# Model accuracy
accuracy_qdatr = mean(pred_train$class == y_train)
accuracy_qdate = mean(pred_test$class == y_test) # 72.85% accuracy
# ErrorRate
error_qdatr = mean(pred_train$class != y_train)
error_qdate = mean(pred_test$class != y_test) # 27.15% error Rate
```

Logistic GAM

```
library(psych)
pairs.panels(lasso_trainx[,5:9],
             smooth = TRUE,      # If TRUE, draws loess smooths
             scale = FALSE,      # If TRUE, scales the correlation text font
             density = TRUE,     # If TRUE, adds density plots and histograms
             ellipses = TRUE,    # If TRUE, draws ellipses
             method = "pearson", # Correlation method (also "spearman" or "kendall")
             pch = 21,           # pch symbol
             lm = FALSE,         # If TRUE, plots linear fit rather than the LOESS (smoothed) fi
t
             cor = TRUE,         # If TRUE, reports correlations
             jiggle = FALSE,     # If TRUE, data points are jittered
             factor = 2,         # Jittering factor
             hist.col = 4,       # Histograms color
             stars = TRUE,       # If TRUE, adds significance level with stars
             ci = TRUE,
             cex.cor=0.5)          # If TRUE, adds confidence intervals
```

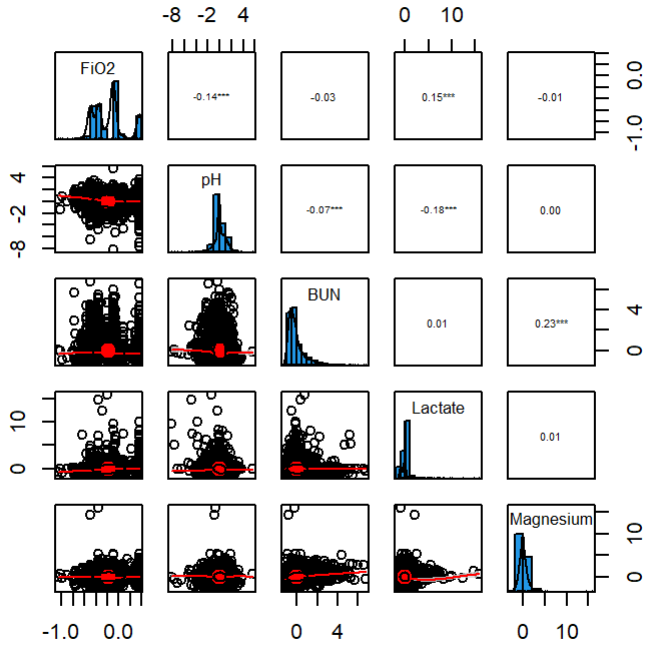

Figure6: GAM Pre-View

```
gam <- gam(as.factor(y_train) ~ s(O2Sat) + s(MAP) +
             s(Resp) + s(FiO2) + s(pH) + s(BUN) +
             s(Lactate) + s(Magnesium) + s(Phosphate) + s(Hgb)+
             s(PTT) + s(WBC) + s(Fibrinogen) + s(Platelets), family=binomial(link='logit'),data=
lasso_trainx)
# Make predictions
pred_train = round(gam%>%predict(as.data.frame(lasso_trainx), type = "response"))
pred_test = round(gam%>%predict(as.data.frame(lasso_testx), type = "response"))
# Model accuracy
accuracy_gamtr = mean(pred_train == as.factor(y_train))
accuracy_gamte = mean(pred_test == as.factor(y_test)) # 71% accuracy
# ErrorRate
error_gamtr = mean(pred_train != as.factor(y_train))
error_gamte = mean(pred_test != as.factor(y_test)) # 29% error Rate
```

KNN Model with CV

```
set.seed(2022)
#Choose the k-fold cross validation of k=5
trControl <- trainControl(method = "cv",
                          number = 5)
train = cbind(y_train,lasso_trainx)
test = cbind(y_test,lasso_testx)
fit <- train(as.factor(SepsisLabel) ~. ,
             method = "knn",
             tuneGrid = expand.grid(k = seq(1:(sqrt(nrow(train))+10))), #The square root of the
 number of observations is 60,therefore, test for 1:65 to test which one fits best
             trControl = trControl,
             metric = "Accuracy",
             data = train)
k <- fit$results%>%filter(Accuracy==max(Accuracy))
k = k[,1] #28
```

```
knn.pred_tain <- knn(train = x_train,
    test = x_train,
    cl = y_train,
    k = k)
tbl_train <- CrossTable(x = knn.pred_tain, y = y_train, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  3645
##
##
##                | y_train
## knn.pred_tain |         0 |         1 | Row Total |
## --------------|-----------|-----------|-----------|
##             0 |      1240 |       572 |      1812 |
##               |     0.684 |     0.316 |     0.497 |
##               |     0.743 |     0.290 |           |
##               |     0.340 |     0.157 |           |
## --------------|-----------|-----------|-----------|
##             1 |       430 |      1403 |      1833 |
##               |     0.235 |     0.765 |     0.503 |
##               |     0.257 |     0.710 |           |
##               |     0.118 |     0.385 |           |
## --------------|-----------|-----------|-----------|
##  Column Total |      1670 |      1975 |      3645 |
##               |     0.458 |     0.542 |           |
## --------------|-----------|-----------|-----------|
##
##
```

```
prop_train <- prop.table(tbl_train[["t"]])
# accuracy
accuracy_knntr <- prop_train[[1]] + prop_train[[4]]

knn.pred_test <- knn(train = x_train,
    test = x_test,
    cl = y_train,
    k = k)
tbl_test <- CrossTable(x = knn.pred_test, y = y_test, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  3028
##
##
##                | y_test
## knn.pred_test |         0 |         1 | Row Total |
## --------------|-----------|-----------|-----------|
##             0 |      1457 |       532 |      1989 |
##               |     0.733 |     0.267 |     0.657 |
##               |     0.826 |     0.421 |           |
##               |     0.481 |     0.176 |           |
## --------------|-----------|-----------|-----------|
##             1 |       307 |       732 |      1039 |
##               |     0.295 |     0.705 |     0.343 |
##               |     0.174 |     0.579 |           |
##               |     0.101 |     0.242 |           |
## --------------|-----------|-----------|-----------|
##  Column Total |      1764 |      1264 |      3028 |
##               |     0.583 |     0.417 |           |
## --------------|-----------|-----------|-----------|
##
##
```

```
prop_test <- prop.table(tbl_test[["t"]])
# accuracy
accuracy_knnte <- prop_test[[1]] + prop_test[[4]]
```

kernel SVM (polynomial Kernel) with CV

https://www.r-bloggers.com/2012/09/learning-kernels-svm/ (https://www.r-bloggers.com/2012/09/learning-kernels-svm/)
https://eight2late.wordpress.com/2017/02/07/a-gentle-introduction-to-support-vector-machines-using-r/ (https://eight2late.wordpress.com/2017/02/07/a-gentle-introduction-to-support-vector-machines-using-r/)
Since tuning gamma, cost, and degree parameters in kernel svm is extremely slow, we only try several parameters and select the best one.

```
# Fit the model
tune_out <- tune.svm(x=lasso_trainx, y=as.factor(y_train),gamma=c(0.01,0.1),cost=c(0.01,0.1),ker
nel="polynomial", degree=c(2,3))
# svm model with gamma0.1, cost0.1, and degree3
svm =svm(as.factor(y_train)~., data = lasso_trainx,  kernel="polynomial",cost=tune_out$best.para
meters$cost,gamma=tune_out$best.parameters$gamma, degree=tune_out$best.parameters$degree)

# Make prediction
pred_train = svm%>%predict(lasso_trainx)
pred_test = svm%>%predict(lasso_testx)
# Model accuracy
accuracy_svmtr = mean(pred_train == y_train)
accuracy_svmte = mean(pred_test == y_test) # 71.1% accuracy
# ErrorRate
error_svmtr = mean(pred_train != y_train)
error_svmte = mean(pred_test != y_test) # 28.90% error Rate
```

XGboost with CV

```
# Convert the dataset to xgb.DMatrix object
x.train = as.matrix(lasso_trainx)
x.test = as.matrix(lasso_testx)

x.train <- Matrix(x.train, sparse = T)
train_data <- list(data=x.train, label = as.integer(y_train))
dtrain <- xgb.DMatrix(data = train_data$data, label = train_data$label)

x.test <- Matrix(x.test, sparse = T)
test_data <- list(data=x.test, label = as.integer(y_test))
dtest <- xgb.DMatrix(data = test_data$data, label = test_data$label)
```

```
# Using for loop
set.seed(2025)
best_param = list()
best_seednumber = 1234
best_logloss = Inf
best_logloss_index = 0
for (iter in 1:100) {
    param <- list(objective = "binary:logistic",
          metrics = "logloss",
          max_depth = sample(2:10, 1),
          eta = runif(1, .01, .5),
          gamma = runif(1, 0.0, 0.2),
          min_child_weight = sample(1:40, 1)
          )
    cv.nround = 100
    cv.nfold = 5
    seed.number = sample.int(10000, 1)[[1]]
    set.seed(seed.number)
    mdcv <- xgb.cv(data=dtrain, params = param,
                   nfold=cv.nfold, nrounds=cv.nround,
                   verbose = T, early_stop_round=20, maximize=FALSE)

    min_logloss = min(mdcv[["evaluation_log"]]$test_logloss_mean)
    min_logloss_index = which.min(mdcv[["evaluation_log"]]$test_logloss_mean)

        if (min_logloss < best_logloss) {
        best_logloss = min_logloss
        best_logloss_index = min_logloss_index
        best_seednumber = seed.number
        best_param = param
        }
}
```

```
# best parameter model
nround = best_logloss_index
set.seed(best_seednumber)
md <- xgboost(data = dtrain, params = best_param, nrounds = nround)
```

```
# Check Accuracy
# train
xgb_tain <- round(predict(md,newdata = dtrain))
tbl_train <- table(as.vector(train_data$label),as.vector(xgb_tain), dnn = c("True","Predict"))
tbl_train <- prop.table(tbl_train)
accuracy_xgbtr <- tbl_train[[1]] + tbl_train[[4]]

# test
xgb_test <- round(predict(md,newdata = dtest))
tbl_test <- table(as.vector(test_data$label),as.vector(xgb_test), dnn = c("True","Predict"))
tbl_test <- prop.table(tbl_test)
accuracy_xgbte <- tbl_test[[1]] + tbl_test[[4]]
```

Model Evaluation

training vs. testing accuracy

```
accurary = data.frame(model = c("LogisticRegression", "LDA", "QDA", "GAM", "KNN", "kernelSVM",
"XGBoost"),
                              train_accuracy=c(accuracy_logtr, accuracy_ldatr, accuracy_qdatr,
accuracy_gamtr, accuracy_knntr, accuracy_svmtr, accuracy_xgbtr),
                              test_accuracy=c(accuracy_logte, accuracy_ldate, accuracy_qdate,
 accuracy_gamte, accuracy_knnte, accuracy_svmte, accuracy_xgbte))
accurary
```

```
##                model train_accuracy test_accuracy
## 1 LogisticRegression      0.7407407     0.7206077
## 2                LDA      0.7423868     0.7186262
## 3                QDA      0.6946502     0.7285337
## 4                GAM      0.7624143     0.7100396
## 5                KNN      0.7251029     0.7229194
## 6          kernelSVM      0.7443073     0.7110304
## 7            XGBoost      0.8691358     0.6935271
```

AUC comparison

```
library(pROC)
pred_log = round(logistic.tr%>%predict(as.data.frame(lasso_testx), type = "response"))
pred_lda = lda%>%predict(lasso_testx)
pred_qda = qda%>%predict(lasso_testx)
pred_gam = round(gam%>%predict(as.data.frame(lasso_testx), type = "response"))
pred_svm = svm%>%predict(lasso_testx)
pred_xgb = round(predict(md,newdata = dtest))

par(pty="s")
roc_log = roc(y_test, pred_log,
              print.auc=TRUE,print.auc.y=0.6,print.auc.cex=0.5,plot=TRUE,col="pink",alpha=0.8,le
gacy.axes=TRUE,lwd =1,xlab= "Specificity")
roc_lda = roc(y_test, pred_lda$posterior[,2],
              print.auc=TRUE,print.auc.y=0.55,print.auc.cex=0.5,plot=TRUE,col="skyblue",alpha=0.
8,legacy.axes=TRUE,lwd =1,add=TRUE)
roc_qda = roc(y_test, pred_qda$posterior[,2],
              print.auc=TRUE,print.auc.y=0.5,print.auc.cex=0.5,plot=TRUE,col="red",alpha=0.8,leg
acy.axes=TRUE,lwd =1,add=TRUE)
roc_gam = roc(y_test, as.numeric(pred_gam),
              print.auc=TRUE,print.auc.y=0.45,print.auc.cex=0.5,plot=TRUE,col="purple",alpha=0.8
,legacy.axes=TRUE,lwd =1,add=TRUE)
roc_knn = roc(y_test, as.numeric(knn.pred_test),
              print.auc=TRUE,print.auc.y=0.4,print.auc.cex=0.5,plot=TRUE,col="blue",alpha=0.8,le
gacy.axes=TRUE,lwd =1,add=TRUE)
roc_svm = roc(y_test, as.numeric(pred_svm),
              print.auc=TRUE,print.auc.y=0.35,print.auc.cex=0.5,plot=TRUE,col="green",alpha=0.8,
legacy.axes=TRUE,lwd =1,add=TRUE)
roc_xgb = roc(y_test, pred_xgb,
              print.auc=TRUE,print.auc.y=0.3,print.auc.cex=0.5,plot=TRUE,col="yellow",alpha=0.8,
legacy.axes=TRUE,lwd =1,add=TRUE)
legend("bottomright",legend=c("LogReg", "LDA", "QDA", "GAM", "KNN", "SVM", "XGBoost"),
       col=c("pink","skyblue","red","purple","blue","green", "yellow"),lwd=2,cex=0.4)
```

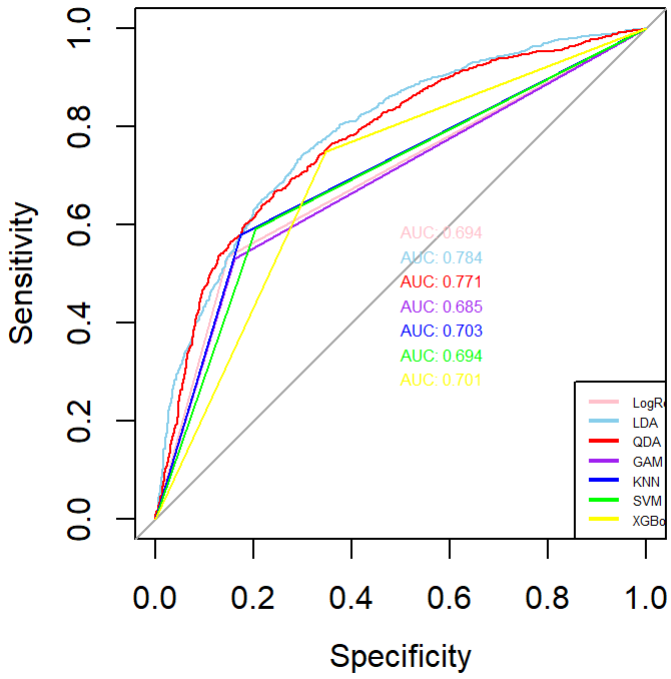

Figure7: ROC-AUC Comparison