Mimic_Proj_PnuemoniaInfluenza

kd91

5/15/2020

Importing Data into R

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 3.6.2
```

```
## Loading required package: ggplot2
```

```
library(ggplot2)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(imputeMissings)
library(forcats)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-2
```

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(rpart.plot)
```

```
## Loading required package: rpart
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

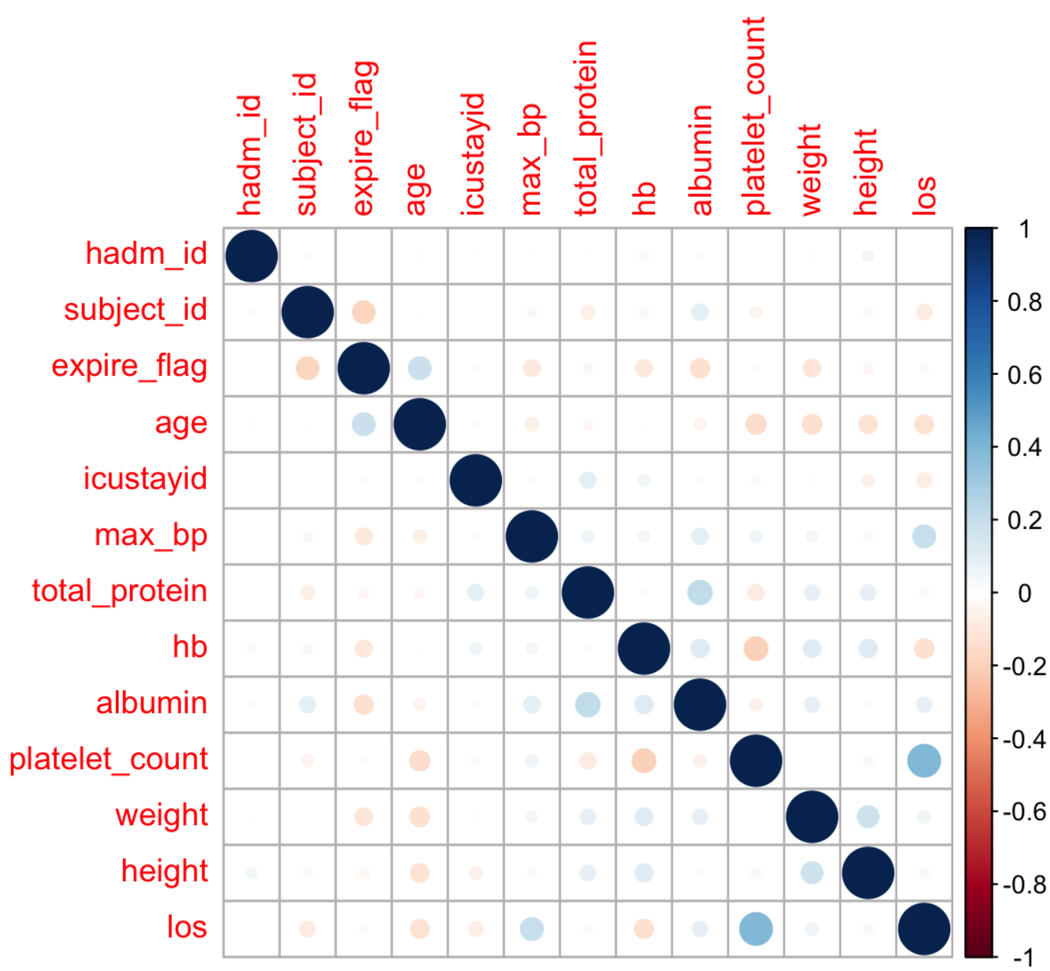
```
setwd(getwd())

df0 <- read.csv(file="data_pneumoniacohort.csv", header=TRUE)
```

EDA

```
# copying df for modifying and analyzing
df1 = df0[2:34]

# change below variables to appropriate type
df1$dischtime <- as.POSIXct(df1$dischtime)
df1$diagnosis <- as.character(df1$diagnosis)
df1$first_admittime <- as.POSIXct(df1$first_admittime)
# df1$expire_flag <- factor(df1$expire_flag)
df1$albumin <- as.numeric(df1$albumin)
df1$platelet_count <- as.numeric(df1$platelet_count)

# adding los to df1 from admittime & dischtime
df1$los <- difftime(df1$dischtime, df1$first_admittime, units = c("days"))
df1$los <- round(df1$los, digits = 0)
df1$los <- as.numeric(df1$los)
```

Correlation

```
# correlation withonly numeric variables:
df1_num <- dplyr::select_if(df1, is.numeric)
cor <- cor(df1_num,use = "pairwise.complete.obs", method = "pearson")
corrplot(cor)
```



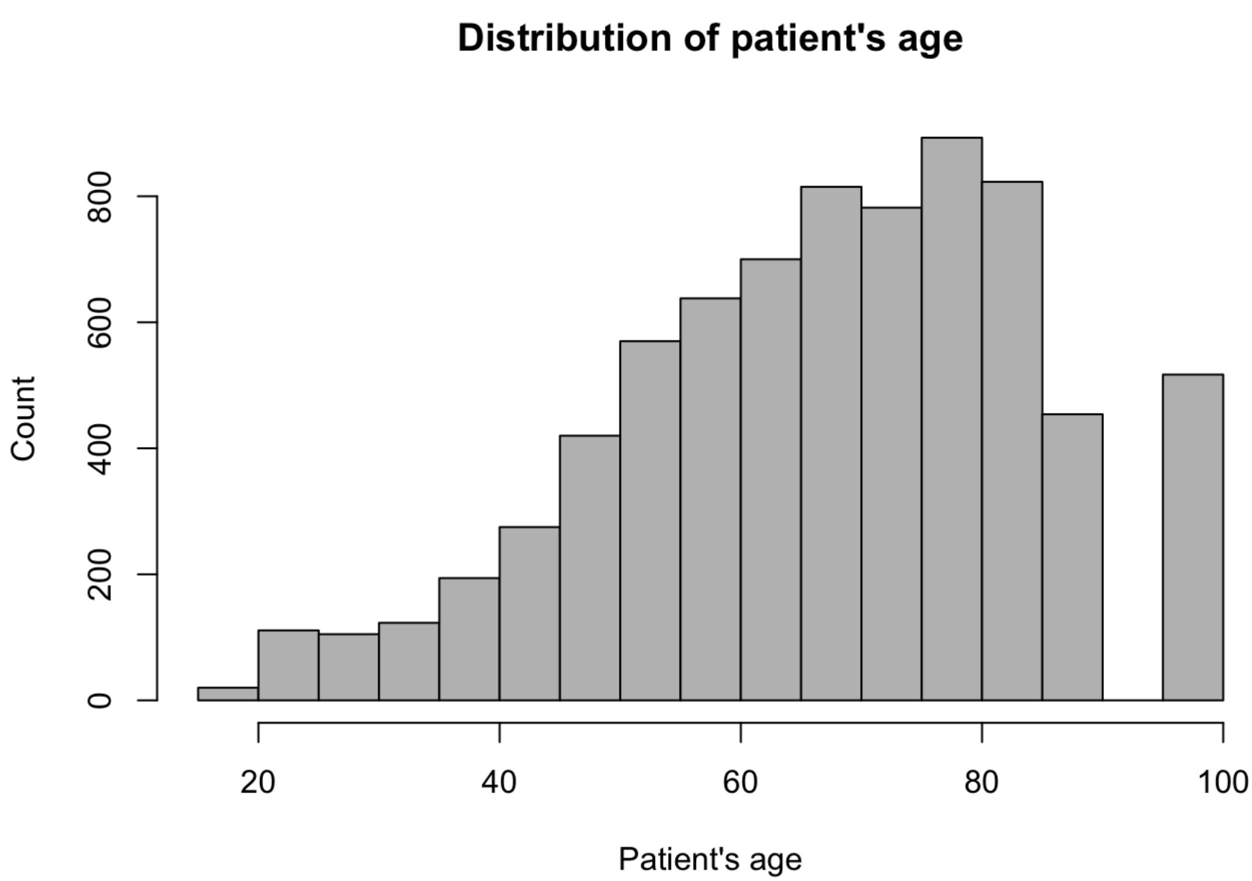
```
# highly positive correlations are seen are between:
# platelet_count & los
# age and expire_flag
# max_bp & los
```

Cleaning impossible values

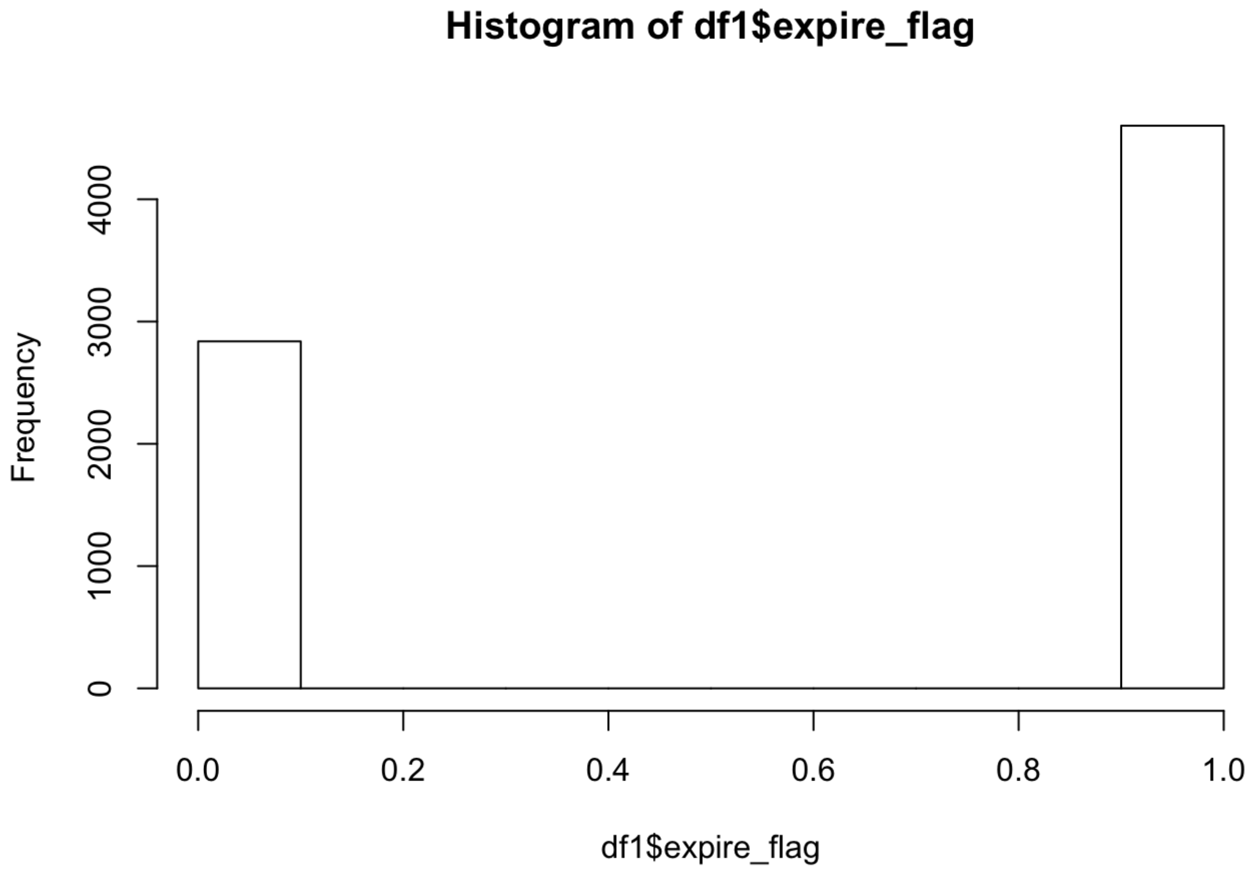
```
# age cannot be more than 110, making these age values to 99
df1$age <- ifelse(df1$age > 110, 99,df1$age)
```

Vizualizations

```
hist(df1$age, col="grey",main="Distribution of patient's age", xlab="Patient's ag
e", ylab="Count")
```

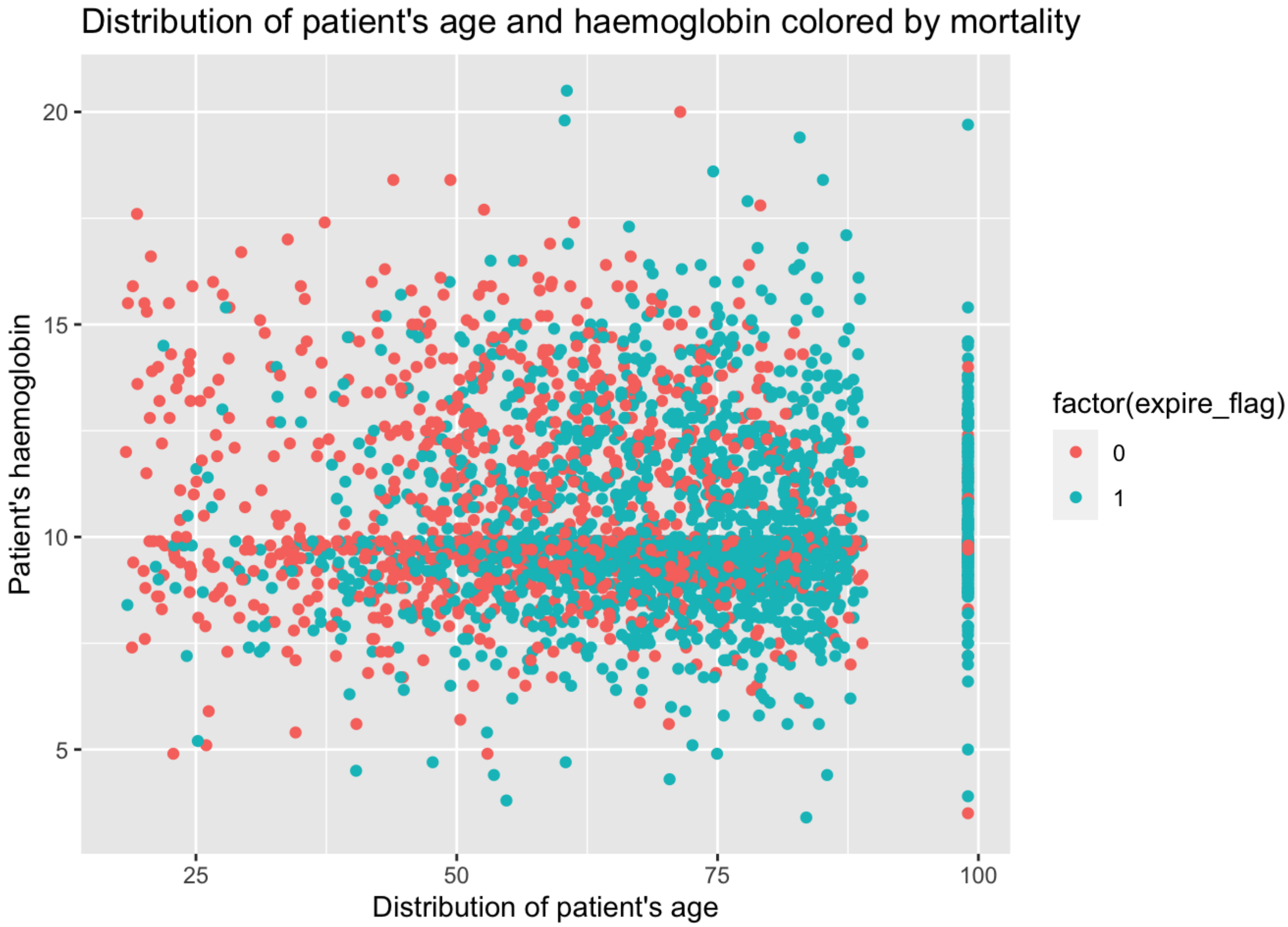



```
hist(df1$expire_flag)
```

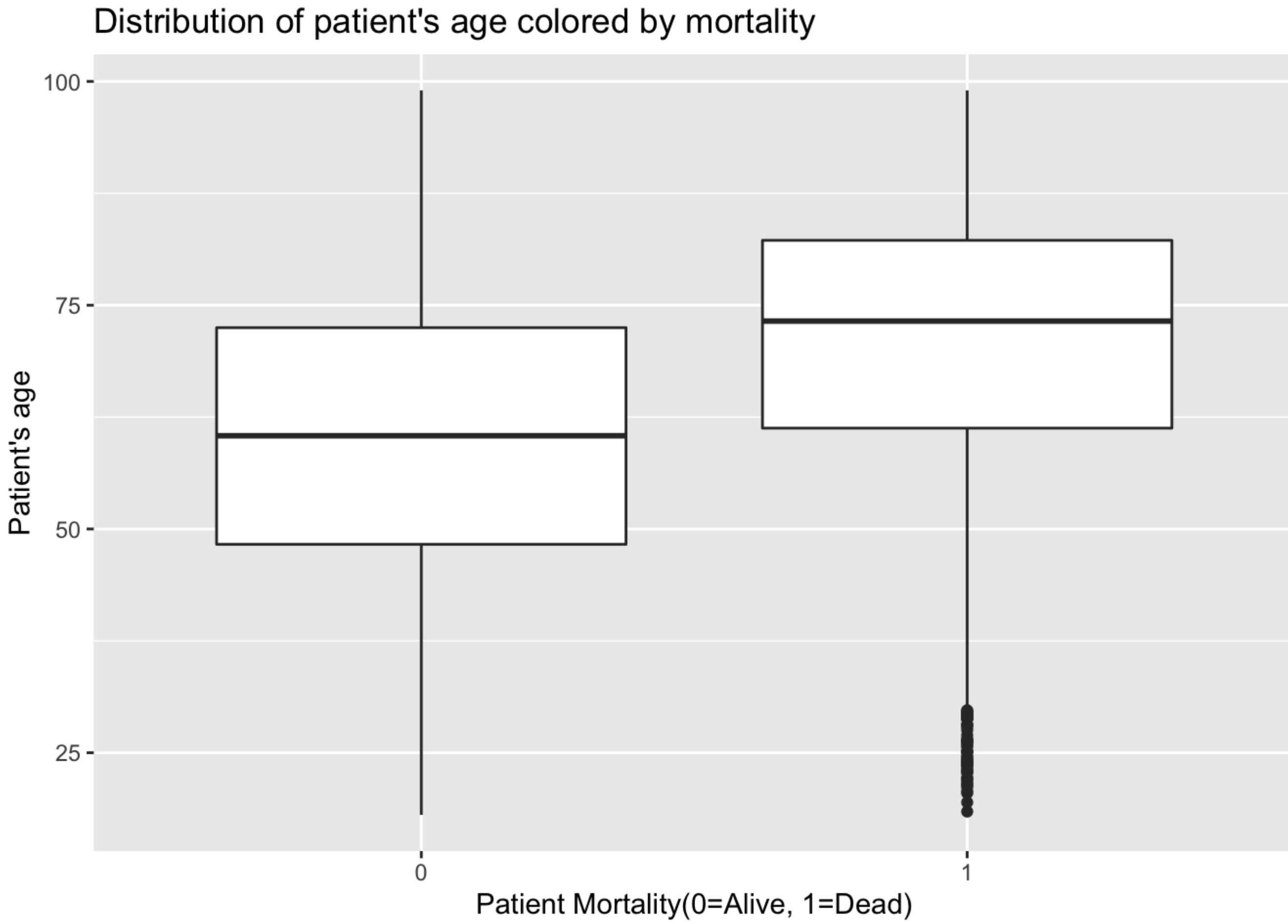



```
# there is a larger populatio of patients that are dead than in the dataset than b
eing alive
# plot indicates no class imbalancein the dataset

g1 = ggplot(df1, aes(x=age,y=hb,col=factor(expire_flag)))
g1 + geom_point()+ ggtitle("Distribution of patient's age and haemoglobin colored
 by mortality") +
  xlab("Distribution of patient's age") + ylab("Patient's haemoglobin")
```

```
## Warning: Removed 4406 rows containing missing values (geom_point).
```

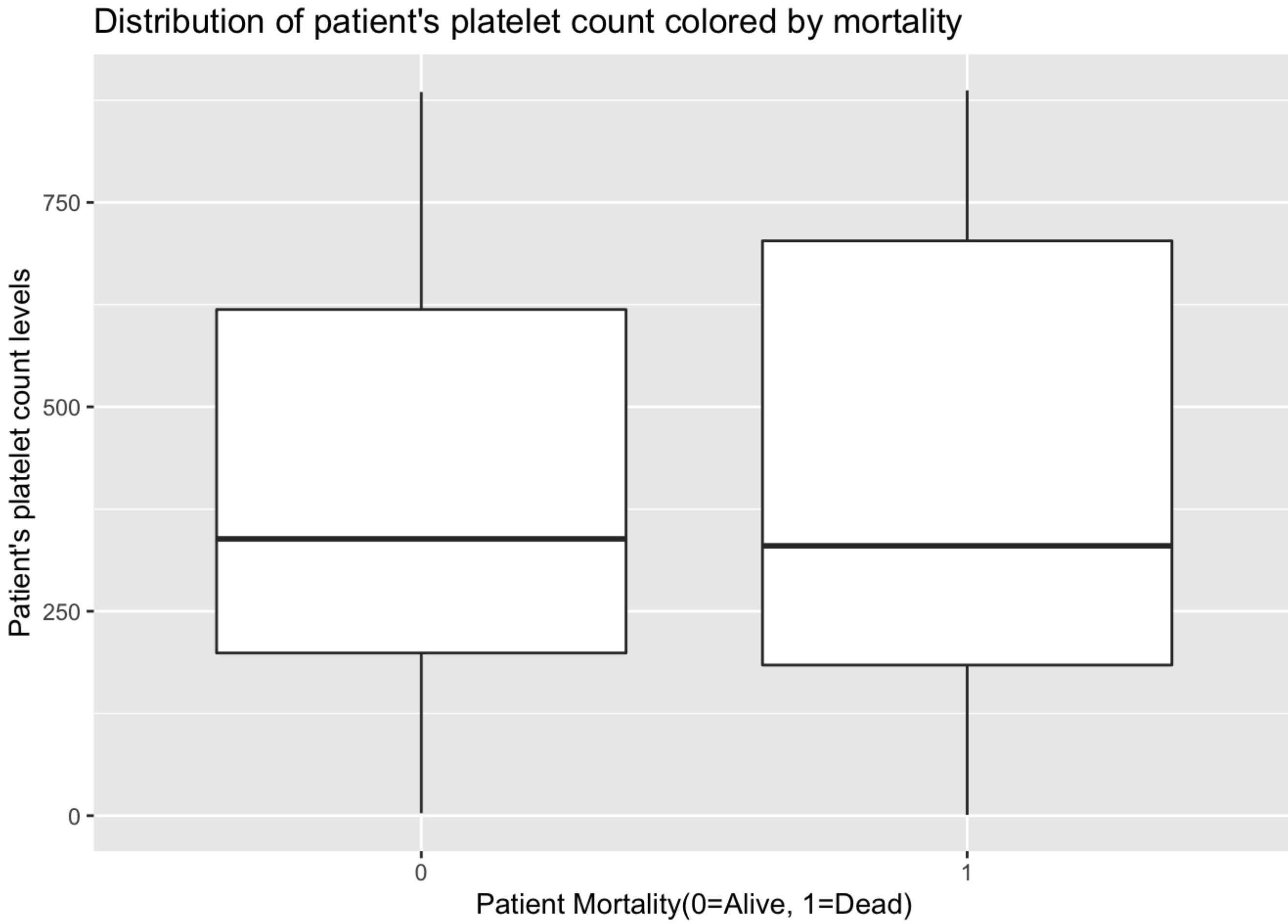



```
# Older patients with less haemoglobin(hb) have higher chance of mortality than yo
unger patients with normal hb levels

g2 = ggplot(df1, aes(x=factor(expire_flag), y=age))
g2 + geom_boxplot()+ ggtitle("Distribution of patient's age colored by mortality")
+
  xlab("Patient Mortality(0=Alive, 1=Dead)") + ylab("Patient's age")
```



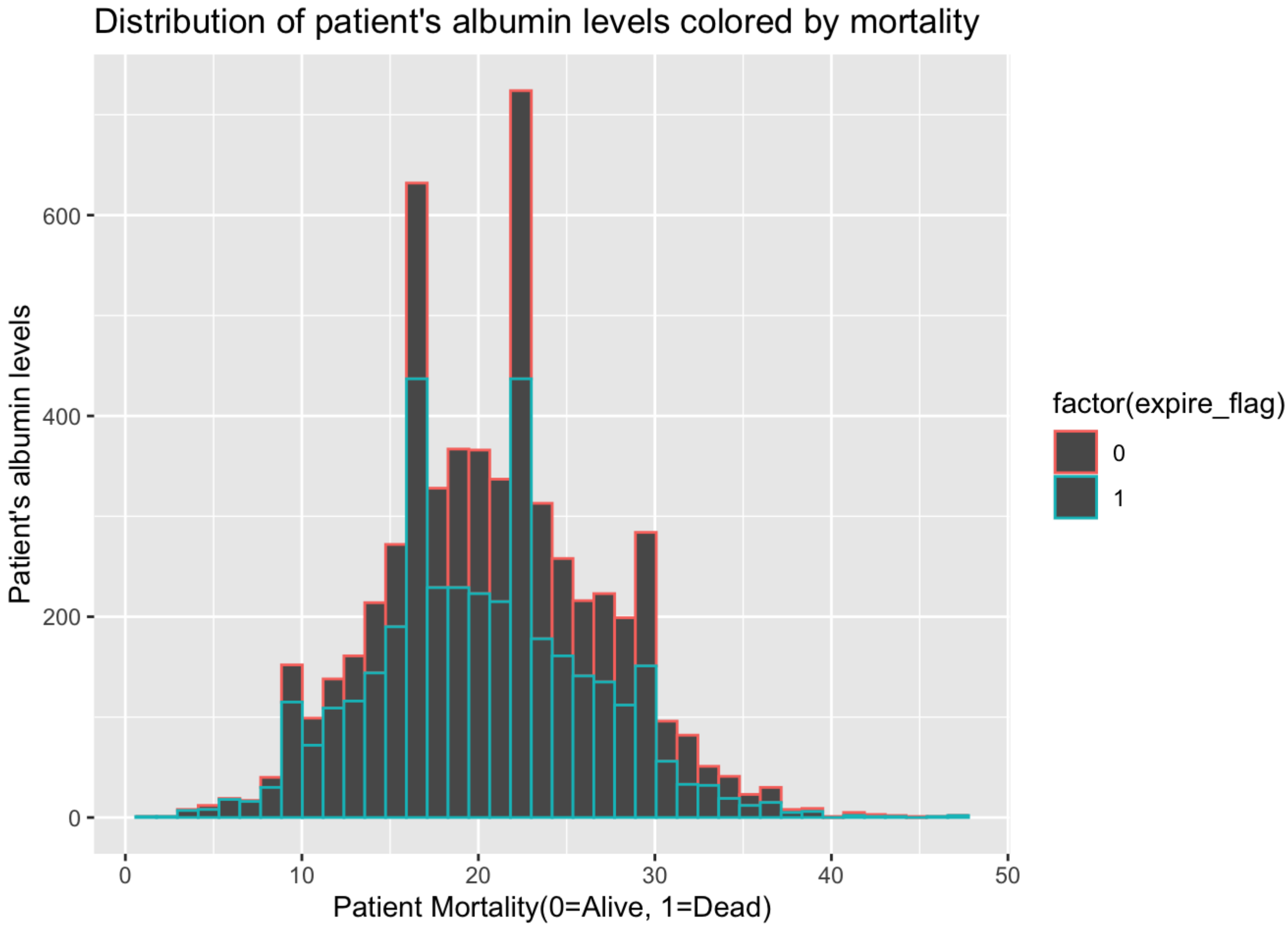

```
# For patients that are dead, the median age is higher than patients that are aliv
e

g3 = ggplot(df1, aes(x=factor(expire_flag), y=platelet_count))
g3 + geom_boxplot()+ ggtitle("Distribution of patient's platelet count colored by
 mortality") +
  xlab("Patient Mortality(0=Alive, 1=Dead)") + ylab("Patient's platelet count leve
ls")
```

```
## Warning: Removed 40 rows containing non-finite values (stat_boxplot).
```

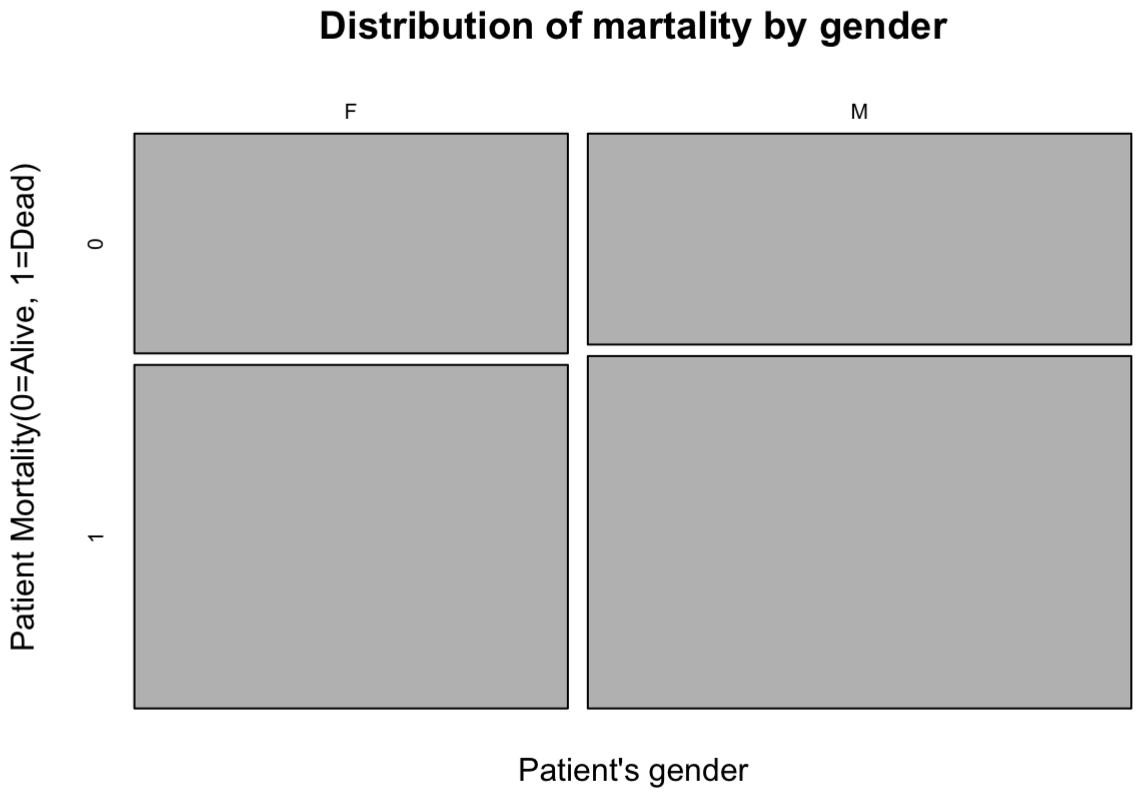



```
# Platelet count of patients did not have much impact on patient's mortality

g4 = ggplot(df1, aes(x=albumin, col=factor(expire_flag)))
g4 + geom_histogram(bins=40)+ ggtitle("Distribution of patient's albumin levels co
lored by mortality") +
  xlab("Patient Mortality(0=Alive, 1=Dead)") + ylab("Patient's albumin levels")
```

```
## Warning: Removed 1704 rows containing non-finite values (stat_bin).
```




```
# For patients that didn't survive, the albumin levels observed are higher than ot
her patients

plot(table(df1$gender,df1$expire_flag), main="Distribution of martality by gender"
, xlab="Patient's gender",
     ylab="Patient Mortality(0=Alive, 1=Dead)")
```




```
# Plot indicates that there is not much difference between mortality of patients b
eing either male or female
```

Data Cleaning

```
# remove redundant variables
# hadm_id,icustayid removed since subject_id suffices
# discahrge_location gives same info as expire_flag
# first_admittime,dischtime accounted into los variable
# diagnosis,short_title,long_title accounted for in icd9_list
# firstcareunits removed, since all patients have ICU admissions
df2 <- subset(df1, select = -c(subject_id,hadm_id,discharge_location,first_admitti
me,dischtime,icustayid,
                              diagnosis,short_title,long_title, religion, firstca
reunits))
summary(df2)
```

```
##                    ethnicity        insurance    gender     admission_type
##  WHITE                    :5393   Government: 145   F:3301   ELECTIVE : 399
##  BLACK/AFRICAN AMERICAN: 667   Medicaid  : 632   M:4139   EMERGENCY:6856
##  UNKNOWN/NOT SPECIFIED : 534   Medicare  :4724            URGENT   : 185
##  HISPANIC OR LATINO    : 176   Private   :1891
##  OTHER                 : 148   Self Pay  :  48
##  ASIAN                 : 119
##  (Other)               : 403
##   expire_flag          age           icd9_list        max_bp      o2_saturation
##  Min.   :0.0000   Min.   :18.06   486     :4761   Min.   :29.0   95% :   1
##  1st Qu.:0.0000   1st Qu.:55.38   48241   : 634   1st Qu.:90.0   NA's:7439
##  Median :1.0000   Median :68.53   4821    : 312   Median :96.0
##  Mean   :0.6185   Mean   :67.04   4829    : 208   Mean   :93.1
##  3rd Qu.:1.0000   3rd Qu.:79.63   48283   : 202   3rd Qu.:99.0
##  Max.   :1.0000   Max.   :99.00   481     : 166   Max.   :99.0
##                                   (Other):1157   NA's   :4235
##    peak_flow    total_protein          hb                   crp
##  60     : 114   Min.   : 2.700   Min.   : 3.40   GREATER THAN 300:  33
##  70     :  99   1st Qu.: 5.000   1st Qu.: 9.20   GREATER THAN 30 :  12
##  80     :  69   Median : 5.600   Median : 9.80   58.8            :   4
##  50     :  60   Mean   : 5.707   Mean   :10.43   20.0            :   3
##  65     :  58   3rd Qu.: 6.300   3rd Qu.:11.70   256.3           :   3
##  (Other): 146   Max.   :13.200   Max.   :20.50   (Other)         : 503
##  NA's   :6894   NA's   :6475     NA's   :4406    NA's            :6882
##     albumin      albumin_urine  platelet_count  creatinine_urine urea_nitrogen
##  Min.   : 1.00   <0.3   :   3   Min.   :  1.0   67     :  43     415    :   8
##  1st Qu.:17.00   0.3    :   3   1st Qu.:191.0   75     :  42     608    :   8
##  Median :21.00   2.4    :   3   Median :332.0   85     :  41     761    :   8
##  Mean   :20.82   4.6    :   3   Mean   :413.5   69     :  40     300    :   7
##  3rd Qu.:25.00   1.1    :   2   3rd Qu.:669.0   84     :  40     485    :   7
##  Max.   :47.00   (Other):  76   Max.   :887.0   (Other):3137     (Other):2193
##  NA's   :1704    NA's   :7350   NA's   :40      NA's   :4097     NA's   :5209
##      weight           height
##  Min.   :  1.00   Min.   :  0.0
##  1st Qu.: 64.72   1st Qu.:160.0
##  Median : 76.30   Median :170.0
##  Mean   : 81.61   Mean   :168.2
##  3rd Qu.: 92.00   3rd Qu.:178.0
##  Max.   :965.50   Max.   :249.0
##  NA's   :4222     NA's   :5497
##                                                         comorbidities
##  PNEUMONIA;CONGESTIVE HEART FAILURE                            :  11
##  CONGESTIVE HEART FAILURE;PNEUMONIA                            :   9
##  CONGESTIVE HEART FAILURE-PNEUMONIA                            :   2
##  RULE-OUT MYOCARDIAL INFARCTION;TELEMETRY;PNEUMONIA            :   2
##  ACUTE MYOCARDIAL INFARCTION;PNEUMONIA;CONGESTIVE HEART FAILURE:   1
##  (Other)                                                       :  28
##  NA's                                                          :7387
```

```
##
smoking_history
##  Former user - stopped more than 1 year ago,Former user - stopped more than 1 y
ear ago                                          : 171
##  Never used,Never used
: 151
##  Former user - stopped more than 1 year ago
: 133
##  Never used
: 125
##  Former user - stopped more than 1 year ago,Former user - stopped more than 1 y
ear ago,Former user - stopped more than 1 year ago: 110
##  (Other)
: 885
##  NA's
:5865
##       los
##  Min.   :  0.00
##  1st Qu.:  6.00
##  Median : 11.00
##  Mean   : 15.42
##  3rd Qu.: 20.00
##  Max.   :295.00
##
```

```
# variables to drop: with more than 50% NA's(=3270)
# o2_saturation: NA's : 7439
# peak_flow: NA's : 6894
# total_protein: NA's :6475
## hb: NA's   :4406
# crp: NA's :6882
# albumin_urine: NA's :7350
# urea_nitrogen: NA's :5209
## creatinine_urine: NA's   :4097
# urea_nitrogen: NA's   :5209
# weight: NA's   :4222
# height: NA's   :5497
# comorbidities: NA's:7387
# smoking_history: NA's: :5865
## max_bp : NAs : 4235

# df3 = df2
# df3 <- na.omit(df3[, c(8,12,17)])
# results in only complete cases with 500 rows if non-NA values of hb,creatinine_u
rine and max_bp are included

df2 <- subset(df2, select = -c(o2_saturation,peak_flow,total_protein,hb,crp,albumi
n_urine,
                               urea_nitrogen,creatinine_urine,urea_nitrogen,max_b
p,weight,
                               height,comorbidities,smoking_history))

#combining similar levels into one
levels(df2$ethnicity)
```

```
##  [1] "AMERICAN INDIAN/ALASKA NATIVE"
##  [2] "AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGNIZED TRIBE"
##  [3] "ASIAN"
##  [4] "ASIAN - ASIAN INDIAN"
##  [5] "ASIAN - CAMBODIAN"
##  [6] "ASIAN - CHINESE"
##  [7] "ASIAN - FILIPINO"
##  [8] "ASIAN - OTHER"
##  [9] "ASIAN - VIETNAMESE"
## [10] "BLACK/AFRICAN"
## [11] "BLACK/AFRICAN AMERICAN"
## [12] "BLACK/CAPE VERDEAN"
## [13] "BLACK/HAITIAN"
## [14] "HISPANIC OR LATINO"
## [15] "HISPANIC/LATINO - CENTRAL AMERICAN (OTHER)"
## [16] "HISPANIC/LATINO - COLOMBIAN"
## [17] "HISPANIC/LATINO - CUBAN"
## [18] "HISPANIC/LATINO - DOMINICAN"
## [19] "HISPANIC/LATINO - GUATEMALAN"
## [20] "HISPANIC/LATINO - MEXICAN"
## [21] "HISPANIC/LATINO - PUERTO RICAN"
## [22] "HISPANIC/LATINO - SALVADORAN"
## [23] "MIDDLE EASTERN"
## [24] "MULTI RACE ETHNICITY"
## [25] "NATIVE HAWAIIAN OR OTHER PACIFIC ISLANDER"
## [26] "OTHER"
## [27] "PATIENT DECLINED TO ANSWER"
## [28] "PORTUGUESE"
## [29] "UNABLE TO OBTAIN"
## [30] "UNKNOWN/NOT SPECIFIED"
## [31] "WHITE"
## [32] "WHITE - BRAZILIAN"
## [33] "WHITE - EASTERN EUROPEAN"
## [34] "WHITE - OTHER EUROPEAN"
## [35] "WHITE - RUSSIAN"
```

```
levels(df2$ethnicity)[c(1:2,23:30)] <- "OTHER"
levels(df2$ethnicity)[2:8] <- "ASIAN"
levels(df2$ethnicity)[3:6] <- "BLACK/AFRICAN AMERICAN"
levels(df2$ethnicity)[4:12] <- "HISPANIC/LATINO"
levels(df2$ethnicity)[5:9] <- "WHITE"

# icd9 codes details:
# http://www.icd9data.com/2015/Volume1/460-519/480-488/default.htm
levels(df2$icd9_list)
```

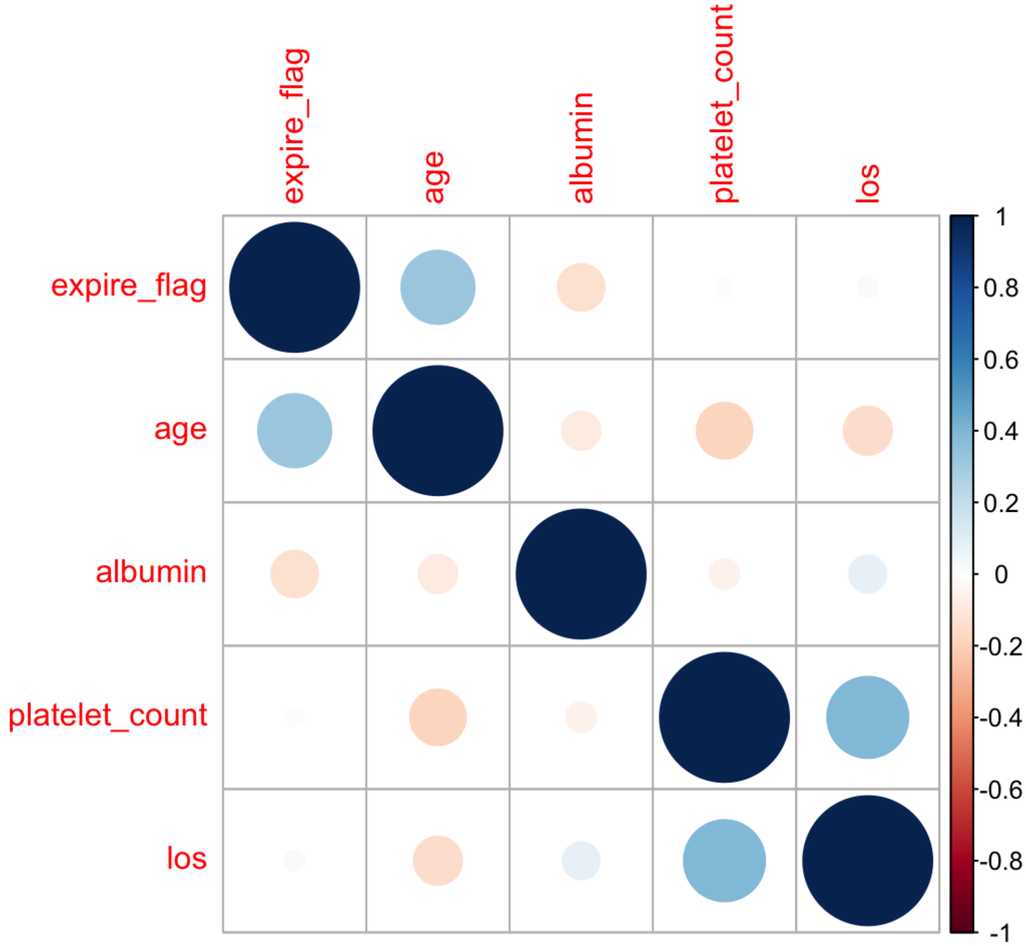
```
##   [1] "4801"                "4801,48241"          "4801,4829"
##   [4] "4802"                "4808"                "4808,48283,486,48241"
##   [7] "4809"                "4809,4830"           "4809,4870"
##  [10] "481"                 "481,4821"            "481,4822"
##  [13] "481,48241"           "481,48242"           "481,48242,486,4822"
##  [16] "481,48283"           "4820"                "4820,481"
##  [19] "4820,4821"           "4820,48240"          "4820,48241"
##  [22] "4820,48242"          "4820,48282"          "4820,48283"
##  [25] "4820,48283,4809"     "4820,486"            "4821"
##  [28] "4821,481"            "4821,4820"           "4821,4820,486"
##  [31] "4821,48240,48283"    "4821,48241"          "4821,48241,48282"
##  [34] "4821,48242"          "4821,48242,486"      "4821,48282"
##  [37] "4821,48283"          "4821,4838"           "4821,4841"
##  [40] "4821,4846"           "4821,4847"           "4821,486"
##  [43] "4821,4870"           "4822"                "4822,481"
##  [46] "4822,4820"           "4822,4821"           "4822,48241"
##  [49] "4822,48242"          "4822,486"            "48230"
##  [52] "48230,4821"          "48230,4822"          "48230,48241"
##  [55] "48231"               "48231,481"           "48232"
##  [58] "48232,486"           "48239"               "48239,4820"
##  [61] "48239,48283"         "48240"               "48240,4821"
##  [64] "48240,48283"         "48241"               "48241,481"
##  [67] "48241,4820"          "48241,4821"          "48241,4821,4820"
##  [70] "48241,4821,4870"     "48241,4822"          "48241,48230"
##  [73] "48241,48232"         "48241,48249"         "48241,48282"
##  [76] "48241,48283"         "48241,4838,4846"     "48241,4841"
##  [79] "48241,4846"          "48241,486"           "48241,4870"
##  [82] "48241,4870,4821"     "48242"               "48242,481"
##  [85] "48242,4820"          "48242,4821"          "48242,48241"
##  [88] "48242,48282"         "48242,48282,4846"    "48242,48283"
##  [91] "48242,4829"          "48242,4878"          "48249"
##  [94] "48249,48283,48282"   "48281"               "48282"
##  [97] "48282,4820"          "48282,4821"          "48282,4822"
## [100] "48282,48241"         "48282,48283"         "48283"
## [103] "48283,481"           "48283,4820"          "48283,4821"
## [106] "48283,48240"         "48283,48241"         "48283,48242"
## [109] "48283,48282"         "48283,4846"          "48283,4871"
## [112] "48284"               "48289"               "48289,4820"
## [115] "4829"                "4829,4801"           "4829,4809"
## [118] "4829,48282,481"      "4829,4846,4801,4847" "4829,4848"
## [121] "4829,4870"           "4829,4871"           "4829,4881"
## [124] "4830"                "4830,48242"          "4830,486"
## [127] "4838"                "4838,4820"           "4838,4821"
## [130] "4838,48241"          "4838,48282"          "4841"
## [133] "4841,4829"           "4841,4846,4821"      "4841,486"
## [136] "4843"                "4846"                "4846,4802"
## [139] "4846,4821"           "4846,48241"          "4846,48281"
## [142] "4846,4829"           "4846,486"            "4846,4870"
```

```
## [145] "4847"                "4847,4821"           "4847,48241"
## [148] "4847,48242"          "4848"                "4848,4846,4821"
## [151] "485"                 "486"                 "486,481"
## [154] "486,4820"            "486,4821"            "486,4822"
## [157] "486,48241"           "486,48283,48241"     "486,48284"
## [160] "486,4829"            "486,4846"            "486,4847"
## [163] "486,486"             "486,4870"            "486,4871"
## [166] "486,4881"            "4870"                "4870,4802"
## [169] "4870,4808"           "4870,4809"           "4870,481"
## [172] "4870,4820,48241"     "4870,4821,48282"     "4870,4822,481"
## [175] "4870,48241"          "4870,48283"          "4870,4829"
## [178] "4871"                "4871,48249"          "4871,4829"
## [181] "4871,486"            "48801"               "4881"
## [184] "4881,4808"           "4881,48242"          "4881,48282,4821"
## [187] "4881,486"            "4881,4870"
```

```
df2$icd9_list <- fct_collapse(df2$icd9_list, Viral_pneumonia="486",Methicillin_sus
p_pneumonia_Staph="48241",
                              Pneumonia_Pseudomonas="4821",Bacterial_pneumonia="48
29",
                              Pneumonia_other_gram_neg_bacteria ="48283",Pneumococ
cal_pneumonia = "481",
                              other_level = "Others")
summary(df2)
```

```
##                   ethnicity         insurance    gender     admission_type
##  OTHER                   : 889   Government: 145   F:3301   ELECTIVE : 399
##  ASIAN                   : 182   Medicaid  : 632   M:4139   EMERGENCY:6856
##  BLACK/AFRICAN AMERICAN: 704   Medicare  :4724            URGENT   : 185
##  HISPANIC/LATINO       : 238   Private   :1891
##  WHITE                   :5427   Self Pay  :  48
##
##
##   expire_flag          age                                         icd9_list
##  Min.   :0.0000   Min.   :18.06   Pneumococcal_pneumonia           : 166
##  1st Qu.:0.0000   1st Qu.:55.38   Pneumonia_Pseudomonas            : 312
##  Median :1.0000   Median :68.53   Methicillin_susp_pneumonia_Staph : 634
##  Mean   :0.6185   Mean   :67.04   Pneumonia_other_gram_neg_bacteria: 202
##  3rd Qu.:1.0000   3rd Qu.:79.63   Bacterial_pneumonia              : 208
##  Max.   :1.0000   Max.   :99.00   Viral_pneumonia                  :4761
##                                   Others                           :1157
##     albumin      platelet_count       los
##  Min.   : 1.00   Min.   :  1.0   Min.   :  0.00
##  1st Qu.:17.00   1st Qu.:191.0   1st Qu.:  6.00
##  Median :21.00   Median :332.0   Median : 11.00
##  Mean   :20.82   Mean   :413.5   Mean   : 15.42
##  3rd Qu.:25.00   3rd Qu.:669.0   3rd Qu.: 20.00
##  Max.   :47.00   Max.   :887.0   Max.   :295.00
##  NA's   :1704    NA's   :40
```

```
#correlation for variables within cleaned dataset
cor2 <- cor(df2[c(5:6,8:10)], use = "pairwise.complete.obs", method = "pearson")
corrplot(cor2)
```




```
# highly positive correlations that are seen are between:
# platelet_count & los
# age and expire_flag

df2$expire_flag <- factor(df2$expire_flag)
```

Modeling

Splitting train-test data and imputation of NA’s

```
set.seed(1000)

intrain <- createDataPartition(y = df2$expire_flag, p= 0.7, list = FALSE)
training <- df2[intrain,]
testing <- df2[-intrain,]
dim(intrain); dim(training); dim(testing)
```

```
## [1] 5209    1
```

```
## [1] 5209   10
```

```
## [1] 2231   10
```

```
# impute with median/mode on train data
values <- compute(training)
training_imp <- impute(training,object=values)
#impute on test data
testing_imp <- impute(testing,object=values)

levels(training_imp$expire_flag) <- c("N", "Y")
levels(testing_imp$expire_flag) <- c("N", "Y")
```

Elastic net Regression

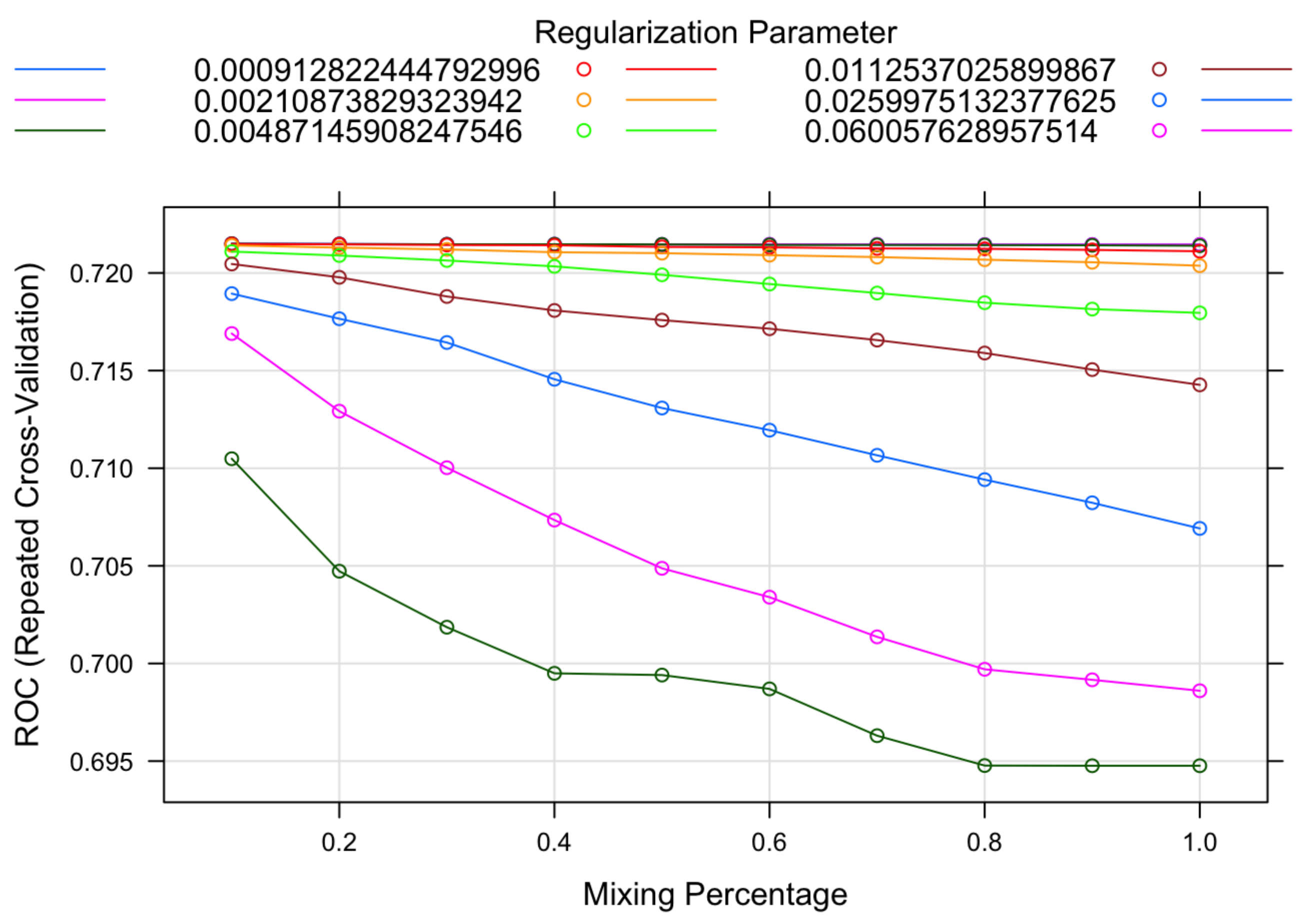
```
set.seed(1001)
trctrl_net <- trainControl(summaryFunction=twoClassSummary,classProbs = TRUE,# Use
AUC to pick the best model
                           method = "repeatedcv", number = 5, repeats = 3)

# grid_net <- expand.grid(alpha = 0:1,lambda = seq(0.0001, 1, length = 20))
# model_net_grid <- train(expire_flag ~., data = training_imp, method = "glmnet",
#                      trControl=trctrl_net, preProcess = c("center", "scale"),
#                     tuneGrid = grid_net,tuneLength = 10)
# better results are obtained without gridsearch

model_net <- train(expire_flag ~., data = training_imp, method = "glmnet",
                   trControl=trctrl_net, preProcess = c("center", "scale"),
                   metric="ROC",tuneLength = 10)
model_net
```

```
## glmnet
##
## 5209 samples
##    9 predictor
##    2 classes: 'N', 'Y'
##
## Pre-processing: centered (21), scaled (21)
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 4166, 4168, 4167, 4168, 4167, 4168, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda        ROC        Sens        Spec
##   0.1    7.404184e-05  0.7214980  0.45228282  0.8387103
##   0.1    1.710463e-04  0.7214980  0.45228282  0.8387103
##   0.1    3.951390e-04  0.7214980  0.45228282  0.8387103
##   0.1    9.128224e-04  0.7214767  0.45228367  0.8390207
##   0.1    2.108738e-03  0.7214114  0.45211701  0.8402622
##   0.1    4.871459e-03  0.7210951  0.45211912  0.8419178
##   0.1    1.125370e-02  0.7204589  0.44641596  0.8441947
##   0.1    2.599751e-02  0.7189420  0.43500711  0.8488499
##   0.1    6.005763e-02  0.7168971  0.41269066  0.8603337
##   0.1    1.387409e-01  0.7104860  0.33988372  0.8909567
##   0.2    7.404184e-05  0.7214806  0.45228325  0.8387103
##   0.2    1.710463e-04  0.7214806  0.45228325  0.8387103
##   0.2    3.951390e-04  0.7214806  0.45228325  0.8387103
##   0.2    9.128224e-04  0.7214643  0.45245159  0.8398483
##   0.2    2.108738e-03  0.7212980  0.45295622  0.8401591
##   0.2    4.871459e-03  0.7208941  0.45228789  0.8415038
##   0.2    1.125370e-02  0.7197776  0.44473754  0.8451253
##   0.2    2.599751e-02  0.7176558  0.43215279  0.8513332
##   0.2    6.005763e-02  0.7129213  0.39793046  0.8652997
##   0.2    1.387409e-01  0.7047258  0.29073242  0.9117518
##   0.3    7.404184e-05  0.7214665  0.45245117  0.8389171
##   0.3    1.710463e-04  0.7214665  0.45245117  0.8389171
##   0.3    3.951390e-04  0.7214665  0.45245117  0.8389171
##   0.3    9.128224e-04  0.7214321  0.45245202  0.8397447
##   0.3    2.108738e-03  0.7212060  0.45362834  0.8407794
##   0.3    4.871459e-03  0.7206393  0.45144784  0.8412974
##   0.3    1.125370e-02  0.7187957  0.44222034  0.8449190
##   0.3    2.599751e-02  0.7164382  0.42678801  0.8521616
##   0.3    6.005763e-02  0.7100264  0.37780042  0.8716111
##   0.3    1.387409e-01  0.7018576  0.23301816  0.9358581
##   0.4    7.404184e-05  0.7214702  0.45261868  0.8388136
##   0.4    1.710463e-04  0.7214702  0.45261868  0.8388136
##   0.4    3.951390e-04  0.7214691  0.45261868  0.8388136
##   0.4    9.128224e-04  0.7214186  0.45228409  0.8397447
##   0.4    2.108738e-03  0.7210618  0.45329333  0.8406760
##   0.4    4.871459e-03  0.7203378  0.45060778  0.8415045
```

```
##   0.4   1.125370e-02  0.7180797  0.44138113  0.8453331
##   0.4   2.599751e-02  0.7145560  0.42225907  0.8526790
##   0.4   6.005763e-02  0.7073441  0.35716365  0.8777152
##   0.4   1.387409e-01  0.6994957  0.18453814  0.9566519
##   0.5   7.404184e-05  0.7214590  0.45278618  0.8389171
##   0.5   1.710463e-04  0.7214590  0.45278618  0.8389171
##   0.5   3.951390e-04  0.7214608  0.45261868  0.8390207
##   0.5   9.128224e-04  0.7213325  0.45261994  0.8397451
##   0.5   2.108738e-03  0.7210148  0.45262247  0.8406760
##   0.5   4.871459e-03  0.7199059  0.45010316  0.8427462
##   0.5   1.125370e-02  0.7175862  0.43886603  0.8461606
##   0.5   2.599751e-02  0.7130839  0.41672426  0.8532991
##   0.5   6.005763e-02  0.7048735  0.34357683  0.8858884
##   0.5   1.387409e-01  0.6994069  0.13420714  0.9691701
##   0.6   7.404184e-05  0.7214631  0.45278618  0.8387104
##   0.6   1.710463e-04  0.7214631  0.45278618  0.8387104
##   0.6   3.951390e-04  0.7214377  0.45261910  0.8392275
##   0.6   9.128224e-04  0.7213123  0.45312330  0.8395383
##   0.6   2.108738e-03  0.7209128  0.45329418  0.8403656
##   0.6   4.871459e-03  0.7194332  0.45010358  0.8424360
##   0.6   1.125370e-02  0.7171451  0.43701800  0.8468845
##   0.6   2.599751e-02  0.7119491  0.40749676  0.8558849
##   0.6   6.005763e-02  0.7033952  0.32445435  0.8984076
##   0.6   1.387409e-01  0.6986981  0.09763680  0.9821022
##   0.7   7.404184e-05  0.7214577  0.45245117  0.8388138
##   0.7   1.710463e-04  0.7214577  0.45245117  0.8388138
##   0.7   3.951390e-04  0.7214341  0.45261910  0.8393309
##   0.7   9.128224e-04  0.7212569  0.45396293  0.8395386
##   0.7   2.108738e-03  0.7208137  0.45346084  0.8403663
##   0.7   4.871459e-03  0.7189720  0.44742056  0.8423326
##   0.7   1.125370e-02  0.7165621  0.43349577  0.8474022
##   0.7   2.599751e-02  0.7106611  0.40011476  0.8585746
##   0.7   6.005763e-02  0.7013625  0.30230498  0.9063736
##   0.7   1.387409e-01  0.6963022  0.05854799  0.9902748
##   0.8   7.404184e-05  0.7214507  0.45278618  0.8387104
##   0.8   1.710463e-04  0.7214507  0.45278618  0.8387104
##   0.8   3.951390e-04  0.7214284  0.45278745  0.8396413
##   0.8   9.128224e-04  0.7212387  0.45429836  0.8399522
##   0.8   2.108738e-03  0.7206806  0.45278956  0.8404698
##   0.8   4.871459e-03  0.7184773  0.44658178  0.8423325
##   0.8   1.125370e-02  0.7158998  0.43114650  0.8467817
##   0.8   2.599751e-02  0.7094154  0.39256568  0.8614709
##   0.8   6.005763e-02  0.6997020  0.28049483  0.9135123
##   0.8   1.387409e-01  0.6947727  0.01408385  0.9980342
##   0.9   7.404184e-05  0.7214540  0.45261868  0.8388139
##   0.9   1.710463e-04  0.7214540  0.45261868  0.8388139
##   0.9   3.951390e-04  0.7214165  0.45278745  0.8397447
##   0.9   9.128224e-04  0.7211815  0.45413254  0.8399520
##   0.9   2.108738e-03  0.7205485  0.45278998  0.8407800
```

```
##   0.9    4.871459e-03  0.7181502  0.44440000  0.8426430
##   0.9    1.125370e-02  0.7150516  0.43031066  0.8479198
##   0.9    2.599751e-02  0.7082297  0.38451240  0.8640580
##   0.9    6.005763e-02  0.6991637  0.26019392  0.9210647
##   0.9    1.387409e-01  0.6947636  0.00000000  1.0000000
##   1.0    7.404184e-05  0.7214535  0.45278618  0.8388139
##   1.0    1.710463e-04  0.7214535  0.45278618  0.8388139
##   1.0    3.951390e-04  0.7213943  0.45245159  0.8397447
##   1.0    9.128224e-04  0.7211121  0.45413212  0.8399515
##   1.0    2.108738e-03  0.7203724  0.45295706  0.8407805
##   1.0    4.871459e-03  0.7179556  0.44456835  0.8425398
##   1.0    1.125370e-02  0.7142697  0.42762638  0.8476096
##   1.0    2.599751e-02  0.7069165  0.37645786  0.8670591
##   1.0    6.005763e-02  0.6986027  0.24241569  0.9286168
##   1.0    1.387409e-01  0.6947636  0.00000000  1.0000000
## 
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.1 and lambda = 0.000395139.
```

```
plot(model_net) # regularization parameter plot
```



```
# standardizes test data the same way as the training data
test_pred_net <- predict(model_net, newdata = testing_imp)

confusionMatrix(test_pred_net, testing_imp$expire_flag, positive="Y")
```

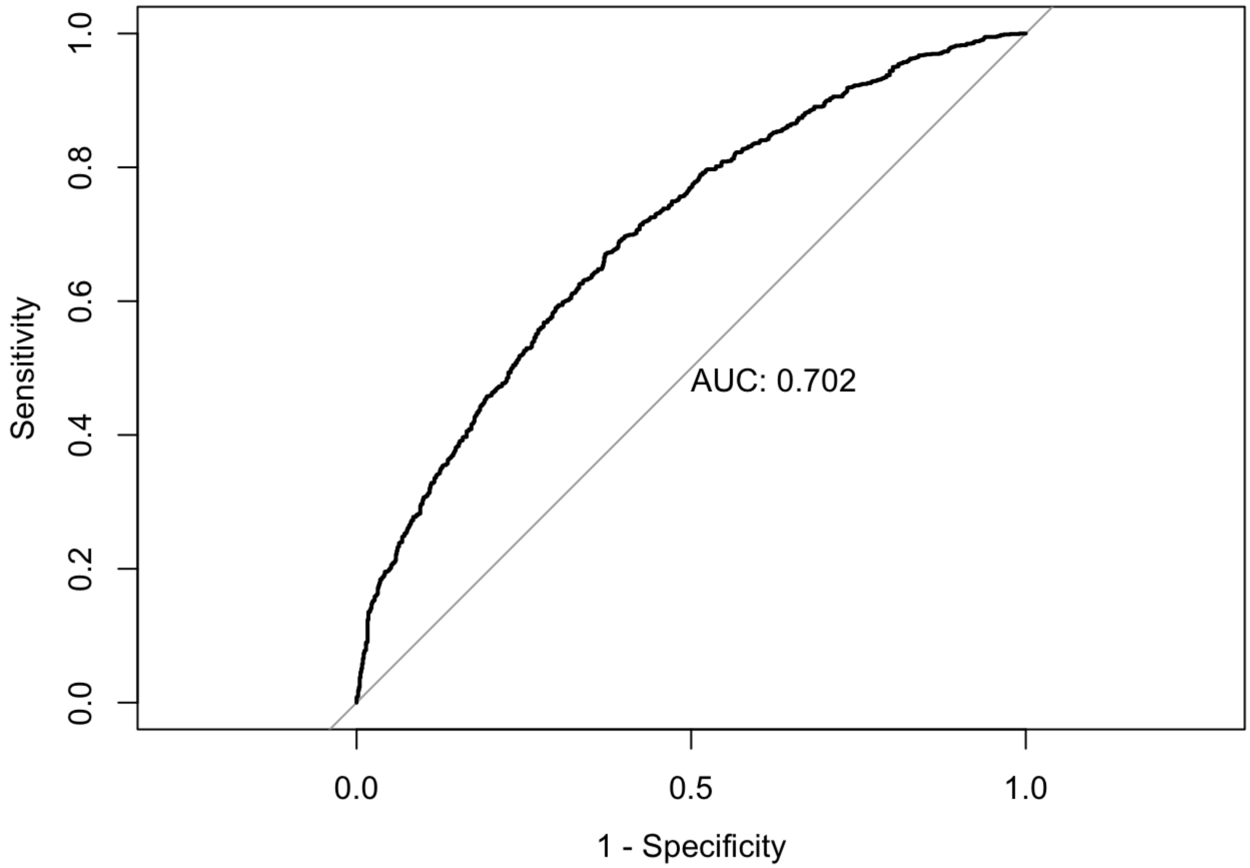
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    Y
##          N  363  245
##          Y  488 1135
##
##                Accuracy : 0.6714
##                  95% CI : (0.6515, 0.6909)
##     No Information Rate : 0.6186
##     P-Value [Acc > NIR] : 1.165e-07
##
##                   Kappa : 0.2634
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8225
##             Specificity : 0.4266
##          Pos Pred Value : 0.6993
##          Neg Pred Value : 0.5970
##              Prevalence : 0.6186
##          Detection Rate : 0.5087
##    Detection Prevalence : 0.7275
##       Balanced Accuracy : 0.6245
##
##        'Positive' Class : Y
##
```

```
rfProbs_net <- predict(model_net, testing_imp, type = "prob")
rfROC_net <- roc(testing_imp$expire_flag, rfProbs_net[, "Y"])
```

```
## Setting levels: control = N, case = Y
```

```
## Setting direction: controls < cases
```

```
plot.roc(rfROC_net, print.auc=TRUE, legacy.axes=TRUE)
```




```
# AUC = 0.702
```

SVM - Linear

```
set.seed(2001)

trctrl_svm <- trainControl(summaryFunction=twoClassSummary,classProbs = TRUE,# Use
AUC to pick the best model
                       method = "repeatedcv", number = 5, repeats = 3)

grid_svm <- expand.grid(C = c(0.005,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5
, 1.75, 2,5))
# grid gave slightly better predictions than the default C values. C=1 when run wi
th defaults

svm_Linear_Grid <- train(expire_flag ~., data = training_imp, method = "svmLinear"
,
                         trControl=trctrl_svm,
                         preProcess = c("center", "scale"),
                         metric = "ROC",tuneLength = 10) # removed tuneGrid = grid
_svm
svm_Linear_Grid
```

```
## Support Vector Machines with Linear Kernel
##
## 5209 samples
##    9 predictor
##    2 classes: 'N', 'Y'
##
## Pre-processing: centered (21), scaled (21)
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 4166, 4167, 4168, 4167, 4168, 4166, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.7123672  0.4757499  0.7888494
##
## Tuning parameter 'C' was held constant at a value of 1
```

```
svm_Linear_Grid$bestTune # C = 1
```

	C <dbl>
1	1

1 row

```
test_pred_svmlinear <- predict(svm_Linear_Grid, newdata = testing_imp)

confusionMatrix(testing_imp$expire_flag, test_pred_svmlinear, positive="Y")
```

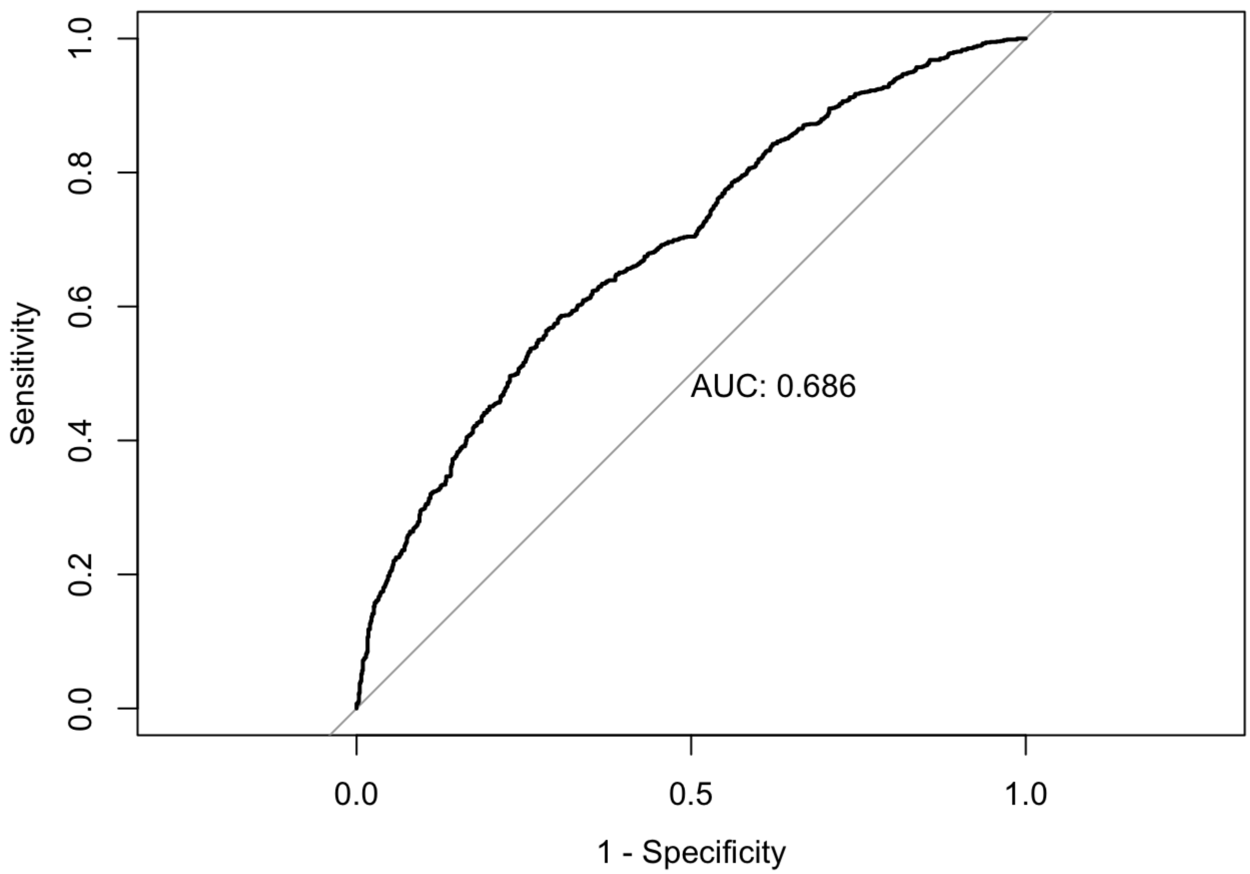
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N 420 431
##          Y 407 973
##
##                Accuracy : 0.6244
##                  95% CI : (0.6039, 0.6445)
##     No Information Rate : 0.6293
##     P-Value [Acc > NIR] : 0.6934
##
##                   Kappa : 0.1997
##
##  Mcnemar's Test P-Value : 0.4269
##
##             Sensitivity : 0.6930
##             Specificity : 0.5079
##          Pos Pred Value : 0.7051
##          Neg Pred Value : 0.4935
##              Prevalence : 0.6293
##          Detection Rate : 0.4361
##    Detection Prevalence : 0.6186
##       Balanced Accuracy : 0.6004
##
##        'Positive' Class : Y
##
```

```
rfProbs_svmlinear <- predict(svm_Linear_Grid, testing_imp, type = "prob")
rfROC_svmlinear <- roc(testing_imp$expire_flag, rfProbs_svmlinear[, "Y"])
```

```
## Setting levels: control = N, case = Y
```

```
## Setting direction: controls < cases
```

```
plot.roc(rfROC_svmlinear, print.auc=TRUE, legacy.axes=TRUE)
```



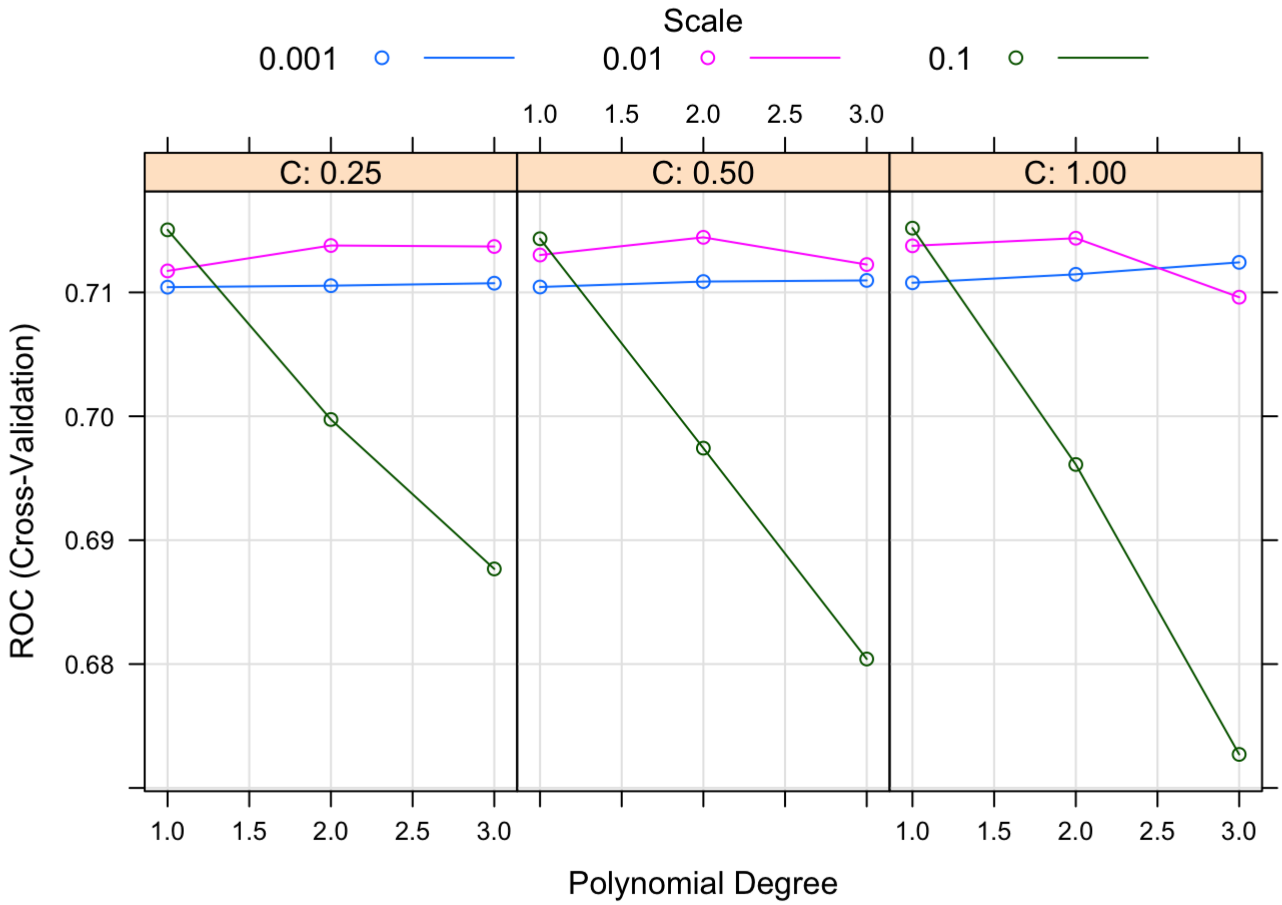

```
# AUC = 0.686
```

SVM - Poly

```
set.seed(2002)

trctrl_svmPoly <- trainControl(summaryFunction=twoClassSummary,classProbs = TRUE,#
Use AUC to pick the best model
                                 method = "cv", number = 3)

# grid search taking too long for svmPoly, runnign model on defaults

svm_Poly <- train(expire_flag ~., data = training_imp, method = "svmPoly",
                         trControl=trctrl_svmPoly,
                         preProcess = c("center", "scale"),
                         metric="ROC",
                         tuneLength = 3)
svm_Poly
```

```
## Support Vector Machines with Polynomial Kernel
##
## 5209 samples
##    9 predictor
##    2 classes: 'N', 'Y'
##
## Pre-processing: centered (21), scaled (21)
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 3472, 3473, 3473
## Resampling results across tuning parameters:
##
##   degree  scale  C     ROC        Sens       Spec
##   1       0.001  0.25  0.7104228  0.5525952  0.7324643
##   1       0.001  0.50  0.7104368  0.5525959  0.7324643
##   1       0.001  1.00  0.7107782  0.5385025  0.7439479
##   1       0.010  0.25  0.7117363  0.5022594  0.7672253
##   1       0.010  0.50  0.7130175  0.4982312  0.7752948
##   1       0.010  1.00  0.7137614  0.5032619  0.7709497
##   1       0.100  0.25  0.7150513  0.4952055  0.7815022
##   1       0.100  0.50  0.7143333  0.4947020  0.7811918
##   1       0.100  1.00  0.7151857  0.4866456  0.7892613
##   2       0.001  0.25  0.7105437  0.5520924  0.7327747
##   2       0.001  0.50  0.7108758  0.5339693  0.7454997
##   2       0.001  1.00  0.7114610  0.5047770  0.7631906
##   2       0.010  0.25  0.7137859  0.4846414  0.7883302
##   2       0.010  0.50  0.7144462  0.4730679  0.8035382
##   2       0.010  1.00  0.7143755  0.4464092  0.8324022
##   2       0.100  0.25  0.6997386  0.3593518  0.8857852
##   2       0.100  0.50  0.6974256  0.3412310  0.8960273
##   2       0.100  1.00  0.6961066  0.3326703  0.8985102
##   3       0.001  0.25  0.7107430  0.5475614  0.7349472
##   3       0.001  0.50  0.7109687  0.5138397  0.7591558
##   3       0.001  1.00  0.7124258  0.4977277  0.7728119
##   3       0.010  0.25  0.7137051  0.4514330  0.8243327
##   3       0.010  0.50  0.7122493  0.4106772  0.8488516
##   3       0.010  1.00  0.7096128  0.3900463  0.8625078
##   3       0.100  0.25  0.6876786  0.3150432  0.8966480
##   3       0.100  0.50  0.6804090  0.2903894  0.9044072
##   3       0.100  1.00  0.6727104  0.2451018  0.9264432
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were degree = 1, scale = 0.1 and C = 1.
```

```
# degree = 1, scale = 0.1 and C = 1.
plot(svm_Poly)
```




```
# standardizes test data the same way as the training data
test_pred_svmPoly <- predict(svm_Poly, newdata = testing_imp)
# test_pred_svmPoly

# API: confusionMatrix(actual, predicted, cutoff = 0.5)
confusionMatrix(testing_imp$expire_flag, test_pred_svmPoly, positive="Y")
```

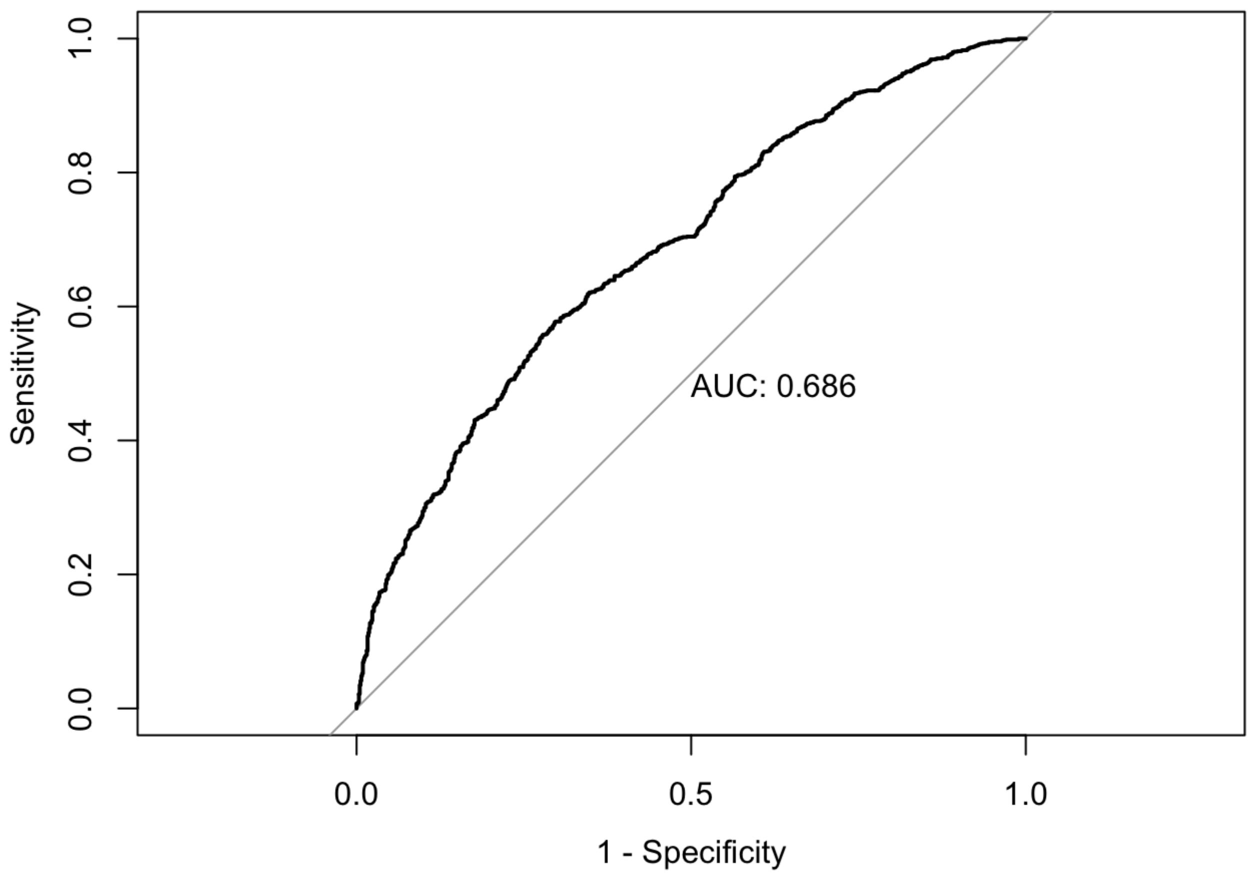
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N 414 437
##          Y 389 991
##
##                Accuracy : 0.6298
##                  95% CI : (0.6093, 0.6498)
##     No Information Rate : 0.6401
##     P-Value [Acc > NIR] : 0.850
##
##                   Kappa : 0.2068
##
##  Mcnemar's Test P-Value : 0.102
##
##             Sensitivity : 0.6940
##             Specificity : 0.5156
##          Pos Pred Value : 0.7181
##          Neg Pred Value : 0.4865
##              Prevalence : 0.6401
##          Detection Rate : 0.4442
##    Detection Prevalence : 0.6186
##       Balanced Accuracy : 0.6048
##
##        'Positive' Class : Y
##
```

```
rfProbs_svmPoly <- predict(svm_Poly, testing_imp, type = "prob")
rfROC_svmPoly <- roc(testing_imp$expire_flag, rfProbs_svmPoly[, "Y"])
```

```
## Setting levels: control = N, case = Y
```

```
## Setting direction: controls < cases
```

```
plot.roc(rfROC_svmPoly, print.auc=TRUE, legacy.axes=TRUE)
```




```
# AUC = 0.686
```

SVM - RBF/Radial

```
set.seed(2033)

trctrl_svmRadial <- trainControl(summaryFunction=twoClassSummary,classProbs = TRUE
,# Use AUC to pick the best model
                                savePredictions = T, method = "repeatedcv", numbe
r = 5)

svmRadialGrid <- expand.grid(sigma= 2^c(-15,-10, -5, 0), C= 2^c(0:5))

svm_Radial_Grid <- train(expire_flag ~., data = training_imp, method = "svmRadial"
,
                         trControl=trctrl_svmRadial,
                         preProcess = c("center", "scale"),
                         metric="ROC",
                         tuneGrid = svmRadialGrid,
                         tuneLength = 10)
```

```
## line search fails -1.018311 -0.07027583 1.078093e-05 9.788953e-06 -2.661898e-08
-3.087078e-10 -2.899992e-13
```

```
## Warning in method$predict(modelFit = modelFit, newdata = newdata, submodels =
## param): kernlab class prediction calculations failed; returning NAs
```

```
## Warning in method$prob(modelFit = modelFit, newdata = newdata, submodels =
## param): kernlab class probability calculations failed; returning NAs
```

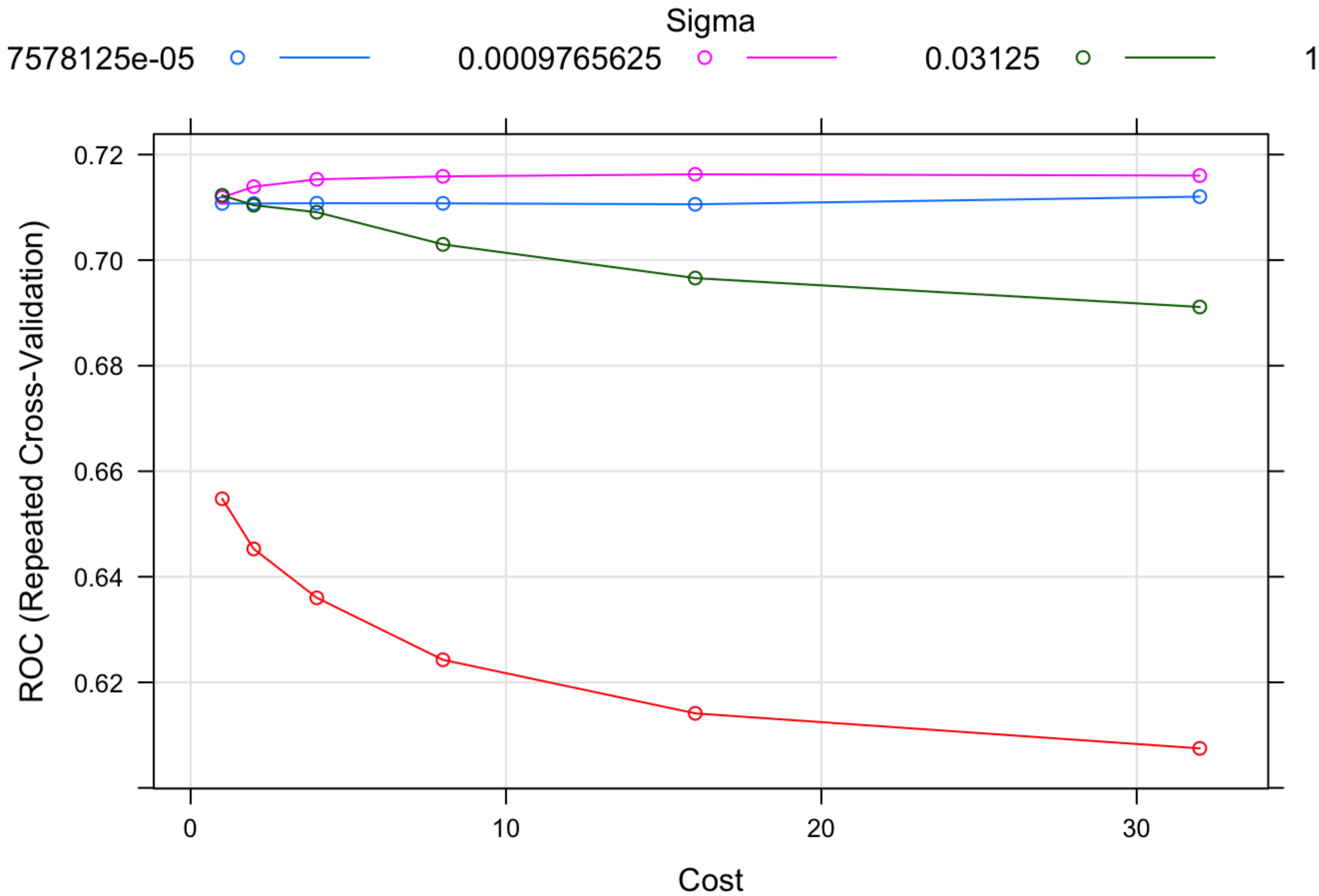
```
## Warning in data.frame(..., check.names = FALSE): row names were found from a
## short variable and have been discarded
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
:
## There were missing values in resampled performance measures.
```

```
svm_Radial_Grid
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 5209 samples
##    9 predictor
##    2 classes: 'N', 'Y'
##
## Pre-processing: centered (21), scaled (21)
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 4168, 4167, 4167, 4167, 4167
## Resampling results across tuning parameters:
##
##   sigma         C   ROC        Sens        Spec
##   3.051758e-05   1  0.7107460  0.55563586  0.7334036
##   3.051758e-05   2  0.7107233  0.55362201  0.7330945
##   3.051758e-05   4  0.7107858  0.55513588  0.7340252
##   3.051758e-05   8  0.7107577  0.55463084  0.7334046
##   3.051758e-05  16  0.7105775  0.52846094  0.7483008
##   3.051758e-05  32  0.7120266  0.50429984  0.7653734
##   9.765625e-04   1  0.7118877  0.50329608  0.7659940
##   9.765625e-04   2  0.7139204  0.49473564  0.7715807
##   9.765625e-04   4  0.7153100  0.49775325  0.7709591
##   9.765625e-04   8  0.7158736  0.49875701  0.7737508
##   9.765625e-04  16  0.7162704  0.47760591  0.7908176
##   9.765625e-04  32  0.7160119  0.45799147  0.8144066
##   3.125000e-02   1  0.7122875  0.39809121  0.8668525
##   3.125000e-02   2  0.7104010  0.39406985  0.8718200
##   3.125000e-02   4  0.7090807  0.38399175  0.8715080
##   3.125000e-02   8  0.7029796  0.37946027  0.8752323
##   3.125000e-02  16  0.6966019  0.38147918  0.8684015
##   3.125000e-02  32  0.6911225  0.35832184  0.8749208
##   1.000000e+00   1  0.6547808  0.25174044  0.8921728
##   1.000000e+00   2  0.6452753  0.24108072  0.8870321
##   1.000000e+00   4  0.6360104  0.20633900  0.8985069
##   1.000000e+00   8  0.6242675  0.17363518  0.9071968
##   1.000000e+00  16  0.6141206  0.10923256  0.9326491
##   1.000000e+00  32  0.6074829  0.07952863  0.9453705
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.0009765625 and C = 16.
```

```
#sigma = 0.0009765625 and C = 16
plot(svm_Radial_Grid)
```




```
# standardizes test data the same way as the training data
test_pred_svmRadial <- predict(svm_Radial_Grid, newdata = testing_imp)

confusionMatrix(testing_imp$expire_flag, test_pred_svmRadial, positive="Y")
```

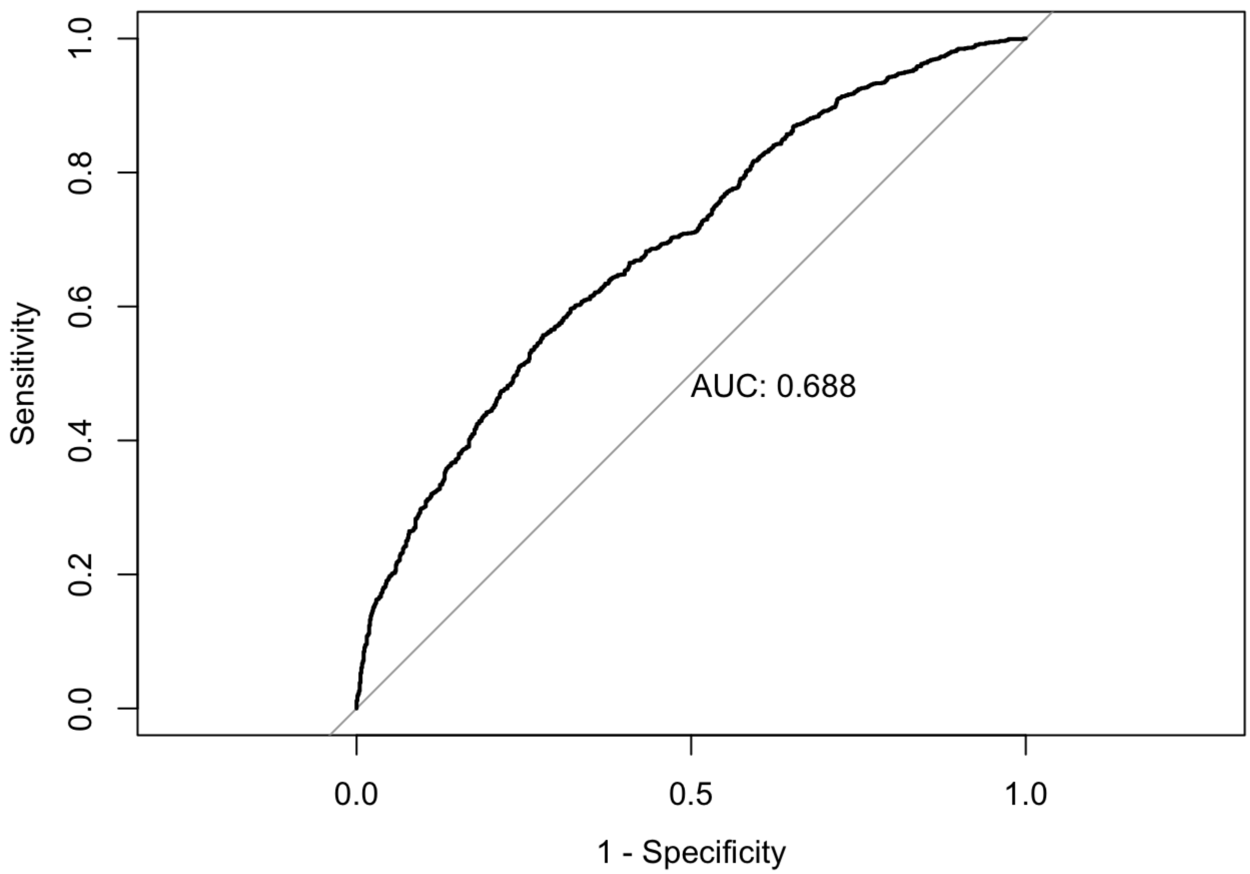
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    Y
##          N  398  453
##          Y  352 1028
##
##                Accuracy : 0.6392
##                  95% CI : (0.6189, 0.6591)
##     No Information Rate : 0.6638
##     P-Value [Acc > NIR] : 0.9933441
##
##                   Kappa : 0.2176
##
##  Mcnemar's Test P-Value : 0.0004242
##
##             Sensitivity : 0.6941
##             Specificity : 0.5307
##          Pos Pred Value : 0.7449
##          Neg Pred Value : 0.4677
##              Prevalence : 0.6638
##          Detection Rate : 0.4608
##    Detection Prevalence : 0.6186
##       Balanced Accuracy : 0.6124
##
##        'Positive' Class : Y
##
```

```
rfProbs_svmRadial <- predict(svm_Radial_Grid, testing_imp, type = "prob")
rfROC_svmRadial <- roc(testing_imp$expire_flag, rfProbs_svmRadial[, "Y"])
```

```
## Setting levels: control = N, case = Y
```

```
## Setting direction: controls < cases
```

```
plot.roc(rfROC_svmRadial, print.auc=TRUE, legacy.axes=TRUE)
```




```
# AUC = 0.688
```

Random Forest

```
set.seed(3011)
trctrl_rf <- trainControl(summaryFunction=twoClassSummary,classProbs = TRUE,# Use
 AUC to pick the best model
                         savePredictions = T,method = "repeatedcv", number = 5, r
epeats = 3)

model_rf <- train(expire_flag ~., data = training_imp, method = "rf",
                  trControl=trctrl_rf,
                  metric="ROC",
                  tuneLength = 10)
model_rf
```

```
## Random Forest
##
## 5209 samples
##    9 predictor
##    2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 4168, 4167, 4167, 4167, 4167, 4167, ...
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##    2    0.7060803  0.4356628  0.8436749
##    4    0.7229908  0.4608289  0.8404686
##    6    0.7237583  0.4775941  0.8280545
##    8    0.7192861  0.4799396  0.8208116
##   10    0.7173831  0.4801041  0.8152253
##   12    0.7149305  0.4784295  0.8121208
##   14    0.7129718  0.4792670  0.8081883
##   16    0.7125582  0.4846407  0.8091208
##   18    0.7112530  0.4784371  0.8087069
##   21    0.7094536  0.4821222  0.8051890
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 6.
```

```
# mtry = 6.

# standardizes test data the same way as the training data
test_pred_rf <- predict(model_rf, newdata = testing_imp)
# test_pred
confusionMatrix(testing_imp$expire_flag, test_pred_rf, positive="Y")
```

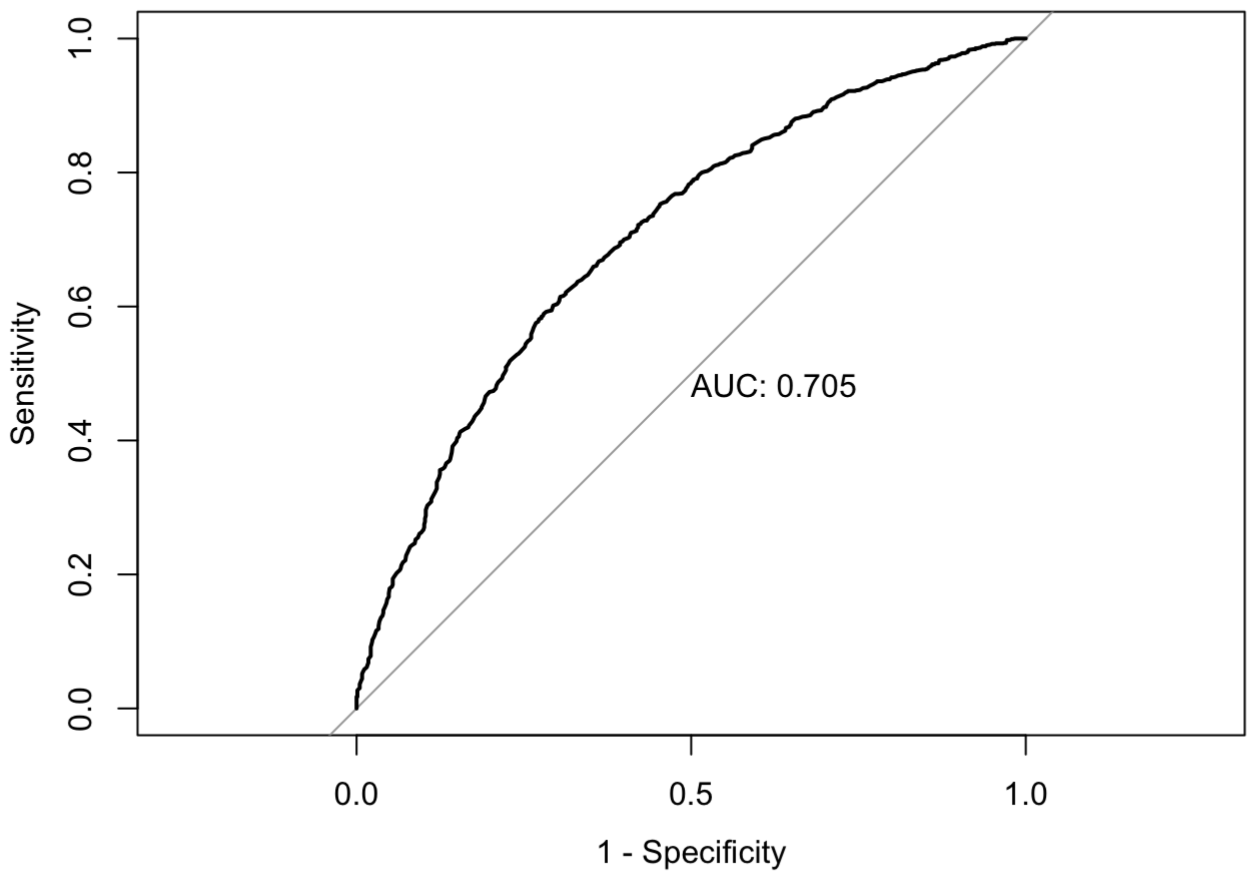
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    Y
##          N  396  455
##          Y  262 1118
##
##                Accuracy : 0.6786
##                  95% CI : (0.6588, 0.698)
##     No Information Rate : 0.7051
##     P-Value [Acc > NIR] : 0.997
##
##                   Kappa : 0.288
##
##  Mcnemar's Test P-Value : 7.479e-13
##
##             Sensitivity : 0.7107
##             Specificity : 0.6018
##          Pos Pred Value : 0.8101
##          Neg Pred Value : 0.4653
##              Prevalence : 0.7051
##          Detection Rate : 0.5011
##    Detection Prevalence : 0.6186
##       Balanced Accuracy : 0.6563
##
##        'Positive' Class : Y
##
```

```
# ROC curve
rfProbs_rf <- predict(model_rf, testing_imp, type = "prob")
# If NAs, can set na.rm=TRUE:
rfROC_rf <- roc(testing_imp$expire_flag, rfProbs_rf[, "Y"])
```

```
## Setting levels: control = N, case = Y
```

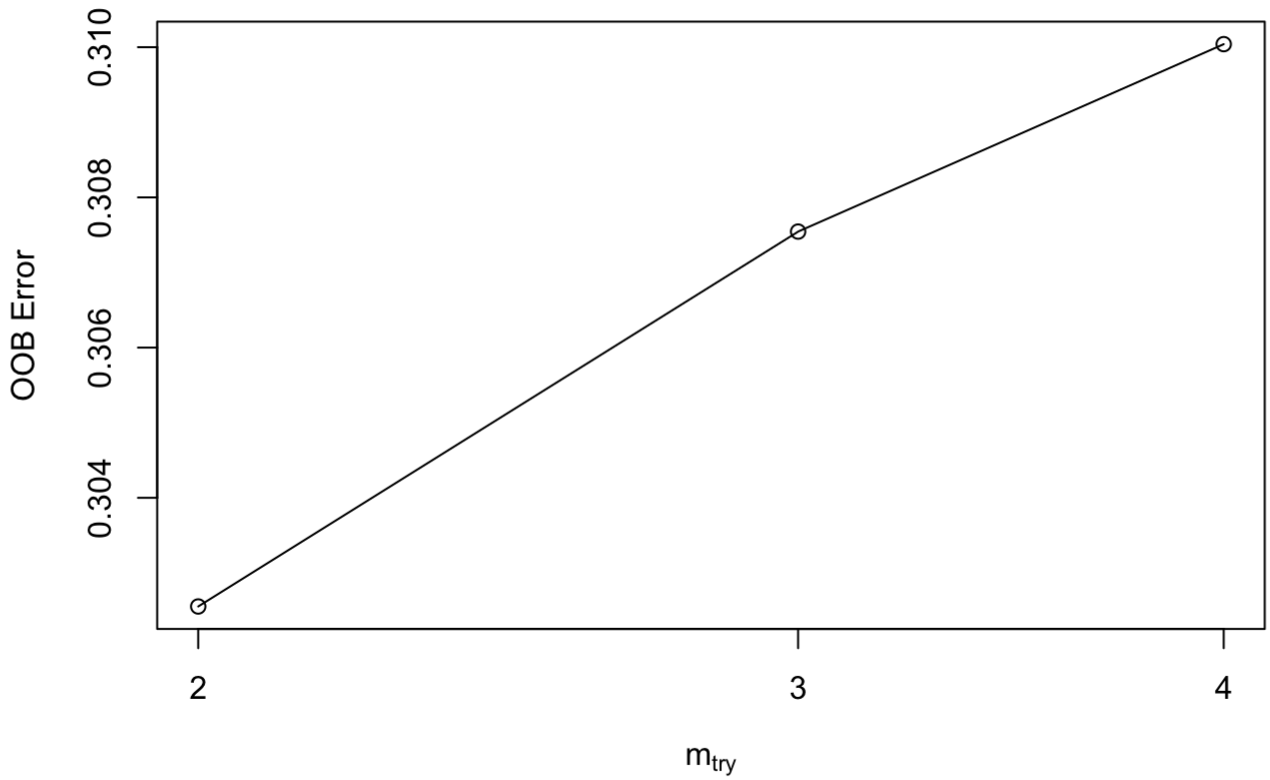
```
## Setting direction: controls < cases
```

```
plot.roc(rfROC_rf, print.auc=TRUE, legacy.axes=TRUE)
```




```
#AUC = 0.705

# ALT: Tune mtry ### better model bestMtry than model_rf
bestMtry <- tuneRF(training_imp[,c(1:4,6:10)], training_imp[,5], stepFactor = 1.5,
improve = 1e-5, ntree = 500, doBest=TRUE)
```

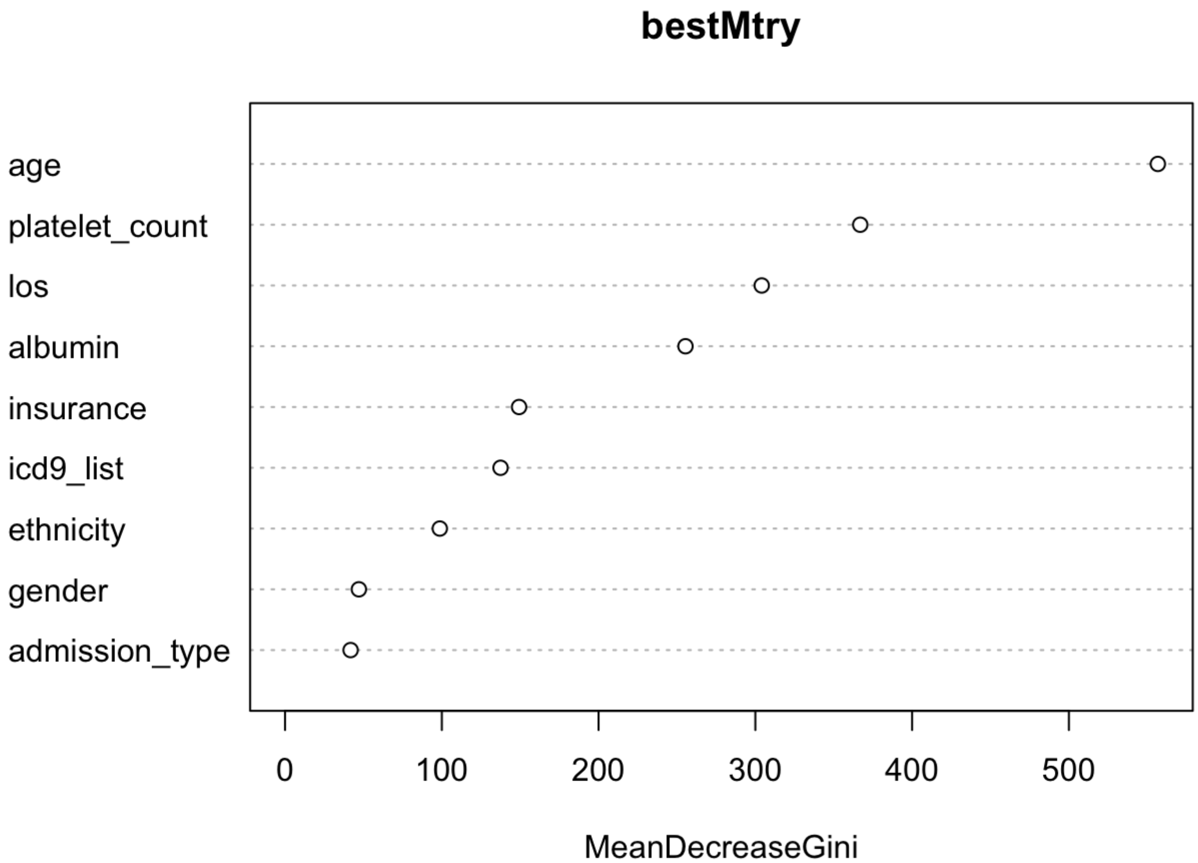
```
## mtry = 3  OOB error = 30.75%
## Searching left ...
## mtry = 2     OOB error = 30.26%
## 0.01622971 1e-05
## Searching right ...
## mtry = 4     OOB error = 31%
## -0.02474619 1e-05
```




```
bestMtry
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = res[which.min(res[, 2]), 1])
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 29.97%
## Confusion matrix:
##     N    Y class.error
## N 944 1043   0.5249119
## Y 518 2704   0.1607697
```

```
# mtry = 2

#importance plot for best rf model
varImpPlot(bestMtry)
```



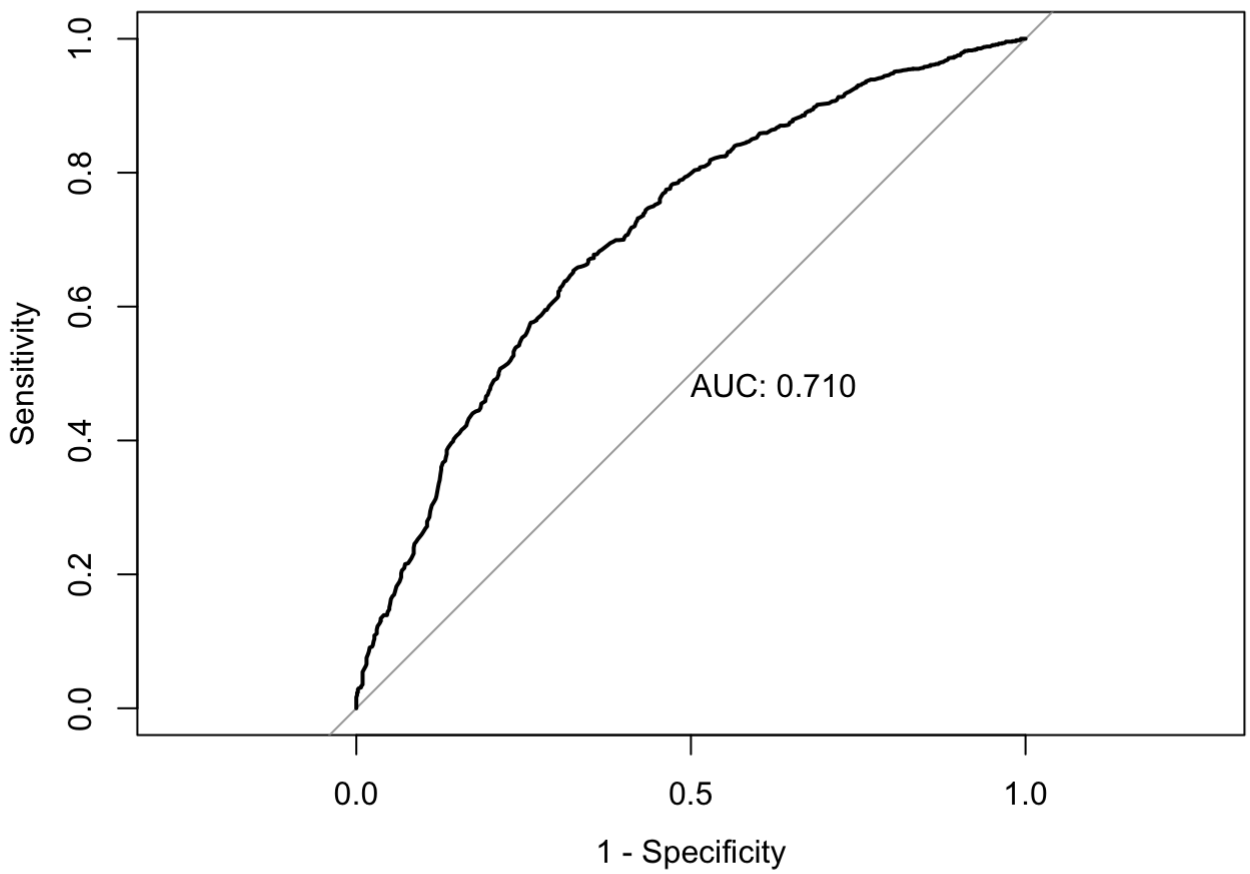

```
# Predict test data after tuning, for confusion matrix:
test_pred_rfbest <- predict(bestMtry, newdata = testing_imp)
#test_pred_rfbest
confusionMatrix(testing_imp$expire_flag, test_pred_rfbest, positive="Y")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    Y
##          N  380  471
##          Y  238 1142
##
##                Accuracy : 0.6822
##                  95% CI : (0.6624, 0.7015)
##     No Information Rate : 0.723
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2892
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7080
##             Specificity : 0.6149
##          Pos Pred Value : 0.8275
##          Neg Pred Value : 0.4465
##              Prevalence : 0.7230
##          Detection Rate : 0.5119
##    Detection Prevalence : 0.6186
##       Balanced Accuracy : 0.6614
##
##        'Positive' Class : Y
##
```

```
# ROC curve
rfProbs_rf <- predict(bestMtry, testing_imp, type = "prob")
# If NAs, can set na.rm=TRUE:
rfROC_rf <- roc(testing_imp$expire_flag, rfProbs_rf[, "Y"])
```

```
## Setting levels: control = N, case = Y
## Setting direction: controls < cases
```

```
plot.roc(rfROC_rf, print.auc=TRUE, legacy.axes=TRUE)
```




```
#AUC = 0.710
```

Conclusions

```
rfROC_net
```

```
##
## Call:
## roc.default(response = testing_imp$expire_flag, predictor = rfProbs_net[,
"Y"])
##
## Data: rfProbs_net[, "Y"] in 851 controls (testing_imp$expire_flag N) < 1380 cas
es (testing_imp$expire_flag Y).
## Area under the curve: 0.7021
```

```
rfROC_svmlinear
```

```
##
## Call:
## roc.default(response = testing_imp$expire_flag, predictor = rfProbs_svmlinear[,
"Y"])
##
## Data: rfProbs_svmlinear[, "Y"] in 851 controls (testing_imp$expire_flag N) < 13
80 cases (testing_imp$expire_flag Y).
## Area under the curve: 0.6862
```

```
rfROC_svmPoly
```

```
##
## Call:
## roc.default(response = testing_imp$expire_flag, predictor = rfProbs_svmPoly[,
"Y"])
##
## Data: rfProbs_svmPoly[, "Y"] in 851 controls (testing_imp$expire_flag N) < 1380
cases (testing_imp$expire_flag Y).
## Area under the curve: 0.6862
```

```
rfROC_svmRadial
```

```
##
## Call:
## roc.default(response = testing_imp$expire_flag, predictor = rfProbs_svmRadial[,
"Y"])
##
## Data: rfProbs_svmRadial[, "Y"] in 851 controls (testing_imp$expire_flag N) < 13
80 cases (testing_imp$expire_flag Y).
## Area under the curve: 0.6876
```

```
rfROC_rf
```

```
##
## Call:
## roc.default(response = testing_imp$expire_flag, predictor = rfProbs_rf[,
"Y"])
##
## Data: rfProbs_rf[, "Y"] in 851 controls (testing_imp$expire_flag N) < 1380 case
s (testing_imp$expire_flag Y).
## Area under the curve: 0.7103
```

```
# The highest to lowest AUC is shown below for all models used above along with ot
her model evaluation metrics:
# 1) Random Forest: AUC = 0.710
                  # Accuracy : 0.6773
                  # Sensitivity : 0.7075
                  # Specificity : 0.6022
# 2) Elastic net: AUC = 0.7021
            # Accuracy : 0.6714
            # Sensitivity : 0.8225
            # Specificity : 0.4266

# 3) SVM - Radial: AUC = 0.6876
                # Accuracy : 0.6392
                # Sensitivity : 0.6941
                # Specificity : 0.5307
# 4) SVM - Poly: AUC = 0.6862
            # Accuracy : 0.6298
            # Sensitivity : 0.6940
            # Specificity : 0.5156
# 5) SVM - Linear: AUC = 0.6862
              # Accuracy : 0.6244
              # Sensitivity : 0.6930
              # Specificity : 0.5079

# The best model is ranfom forest for this dataset, it has better AUC, accuracy an
d good predictions compared to all other models
# Elastic net is second best, however it has very less specificity, which could le
ad to more false positive predictions.
```