WQD7006 Assigment Step 3: Early prediction of Covid-19 ventilation requirement using machine learning

Baharul Hisyam bin Baharudin (S2039609)

Contents

Introduction

The purpose of the study is to use machine learning to predict ventilation requirement of COVID-19 patients using information from baseline chest radiographs, and then evaluate the prediction results.

The research objective is as below:

- To predict ventilatory requirement of COVOD-19 patient based on severity score of baseline chest radiographs
- To compare and evaluate the performance of the predictive model (GLM, SVM, and RF).

For this project, data taken from https://data.mendeley.com/datasets/r6t9tmzzmz/2 will be utilized. The data was collected from confirmed admitted COVID-19 patients (positive PT-PCR test) to King Abdulaziz Medical City in Riyadh by a team of medical researchers in Saudi Arabia. The complete raw dataset contains 36 features:

- 4 General features (Gender, Age at diagnosis, Ventilation support status, vitality status)
- 12 Chest X-Ray features (severity score)
- 10 Complete blood count features
- 10 Comorbidity features

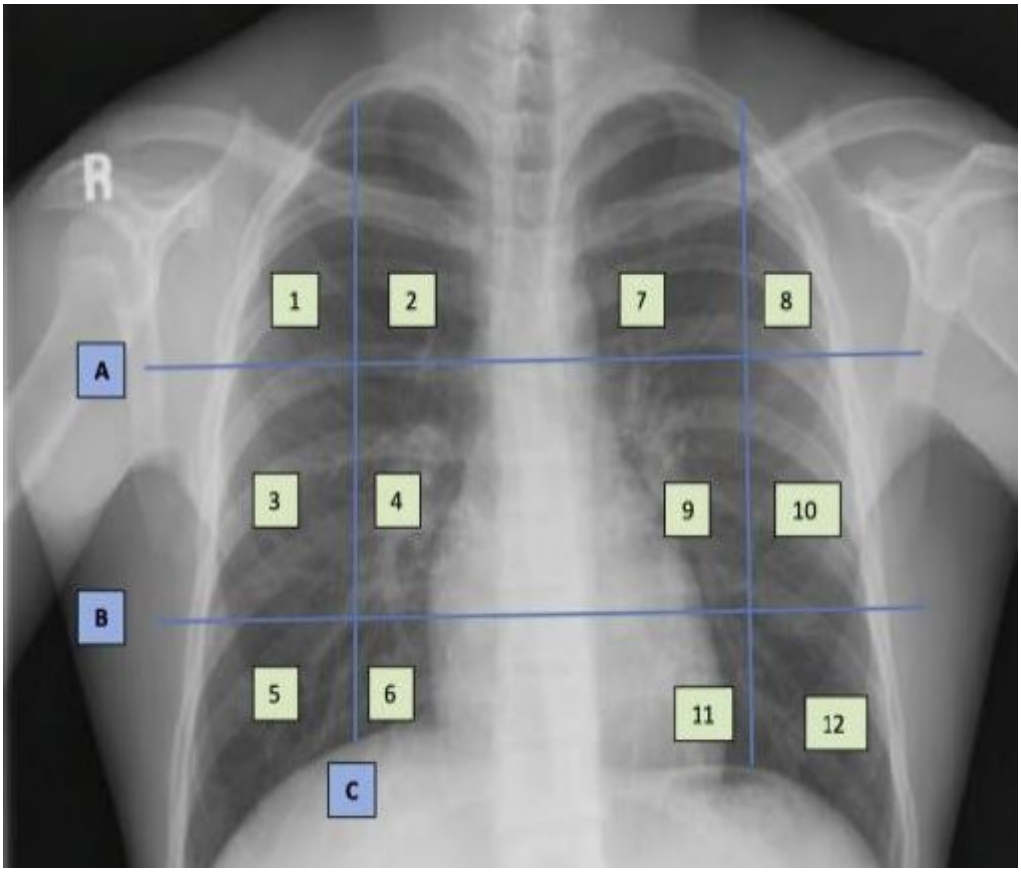


Figure above shows the frontal chest X-ray lung zone segmentation. The horizontal line A and B represent the upper and lower poles of the hilum. The vertical line C is from the junction of the middle/inner third of the clavicle to the diaphragm. the light squares are the regions in which radiologists assign a severity score. Zero is assigned if no manifestation is found, one for mild/moderate, and two for severely affected zones.

Analysis and Design

For this study, R programming language is chosen as the tools together with the caret package (caret is short for classification and regression training). R is chosen due to:

- Its increasing popularity with the universities running statistical and data science courses.
- The availability of pre-compiled libraries and package.
- The fact that R is a free tool.

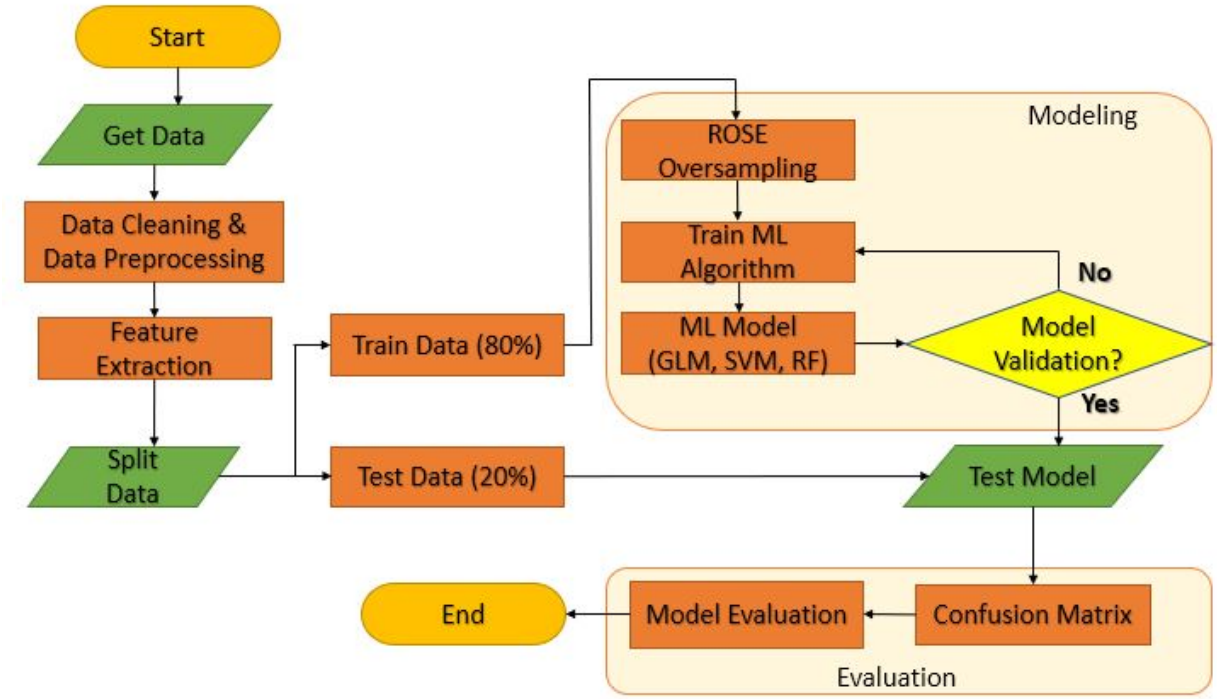


Figure above shows the algorithm (system flow) to process the data according to project objectives.

Coding

```
# load R libraries
library(dplyr) # for data frame manipulation
library(tidyverse) # for data manipulation and visualization
library(ggplot2) # for plotting graphs
library(gridExtra) # for plotting graphs
library(stringr) # for string manipulation
library(caret) # for ML model training and evaluation
library(ROSE) # for synthetic generation of unbalance dataset
library(precrec) # for ML model evaluation
```

Before loading dataset into R, the raw dataset were cleaned first in microsoft excel. The raw dataset has 1513 rows and 37 columns.

- remove ”N/A’ from ventilation.support.status (5 rows).
- remove “deceased” from vital.status (135 rows).
- recode ventilation.support.status to “yes” and “no” to generate binary classification problem.
- remove 10 columns of CBC and 10 columns of comorbidity, since we are only interested in the chest radiographs features and general features.

The dataset is now reduces from 1513 rows x 37 columns to 1373 rows x 17 columns.

```r
# read the dataset
dataset <- read.csv("KAMC_COVID-19 - Copy.csv")
# check dimension of raw dataset
dim(dataset)
```

```
## [1] 1373   17
```

```r
# summary of raw dataset
dataset %>% skimr::skim()
```

Table 1: Data summary

Name	Piped data
Number of rows	1373
Number of columns	17
Column type frequency:	
character	4
numeric	13
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Case.ID	0	1	6	6	0	1373	0
Gender	0	1	4	6	0	2	0
Ventilation.support.status	0	1	2	3	0	2	0
Vital.status	0	1	5	5	0	1	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Age.at.diagnosis	0	1	53.31	16.52	18.77	41.04	53.98	65.33	91.51	
CXR.zone.1	0	1	0.13	0.36	0.00	0.00	0.00	0.00	2.00	
CXR.zone.2	0	1	0.08	0.29	0.00	0.00	0.00	0.00	2.00	
CXR.zone.3	0	1	0.33	0.52	0.00	0.00	0.00	1.00	2.00	
CXR.zone.4	0	1	0.15	0.39	0.00	0.00	0.00	0.00	2.00	
CXR.zone.5	0	1	0.43	0.57	0.00	0.00	0.00	1.00	2.00	
CXR.zone.6	0	1	0.33	0.52	0.00	0.00	0.00	1.00	2.00	
CXR.zone.7	0	1	0.03	0.19	0.00	0.00	0.00	0.00	2.00	
CXR.zone.8	0	1	0.10	0.32	0.00	0.00	0.00	0.00	2.00	
CXR.zone.9	0	1	0.15	0.37	0.00	0.00	0.00	0.00	2.00	
CXR.zone.10	0	1	0.33	0.52	0.00	0.00	0.00	1.00	2.00	
CXR.zone.11	0	1	0.34	0.57	0.00	0.00	0.00	1.00	2.00	
CXR.zone.12	0	1	0.46	0.60	0.00	0.00	0.00	1.00	2.00	

```r
# transfer dataset to transformData variable
transformData <- dataset

# function to 'binned' the age variable
group_age <- function(age){
  if (age > 0 & age < 20){
    return('Below 20 yrs')
  }else if(age >= 20 & age < 30){
    return('20-30 yrs')
  }else if (age >= 30 & age < 40){
    return('30-40 yrs')
  }else if (age >= 40 & age < 50){
    return('40-50 yrs')
  }else if (age >= 50 & age < 60){
    return('50-60 yrs')
  }else if (age >= 60 & age < 70){
    return('60-70 yrs')
  }else if (age >= 70 & age < 80){
    return('70-80 yrs')
  }else if (age >= 80){
    return('Above 80 yrs')
  }
}

# "binned" the age variable into age_group
transformData$age_group <- sapply(transformData$Age.at.diagnosis,group_age)

# convert data types to factors
transformData <-
  transformData %>%
  mutate(Gender = as.factor(Gender)) %>%
  mutate(Ventilation.support.status = as.factor(Ventilation.support.status)) %>%
  mutate(age_group = as.factor(age_group)) %>%
  mutate(CXR.zone.1 = as.factor(CXR.zone.1)) %>%
  mutate(CXR.zone.2 = as.factor(CXR.zone.2)) %>%
  mutate(CXR.zone.3 = as.factor(CXR.zone.3)) %>%
  mutate(CXR.zone.4 = as.factor(CXR.zone.4)) %>%
  mutate(CXR.zone.5 = as.factor(CXR.zone.5)) %>%
  mutate(CXR.zone.6 = as.factor(CXR.zone.6)) %>%
  mutate(CXR.zone.7 = as.factor(CXR.zone.7)) %>%
  mutate(CXR.zone.8 = as.factor(CXR.zone.8)) %>%
  mutate(CXR.zone.9 = as.factor(CXR.zone.9)) %>%
  mutate(CXR.zone.10 = as.factor(CXR.zone.10)) %>%
  mutate(CXR.zone.11 = as.factor(CXR.zone.11)) %>%
  mutate(CXR.zone.12 = as.factor(CXR.zone.12))

# remove unused features
transformData$Case.ID <- NULL # unique case id, not used
transformData$Vital.status <- NULL # only 1 value (Alive), not used
transformData$Age.at.diagnosis <- NULL # variable recoded in age_group

# check new dimension of transformed data
dim(transformData)
```

```
## [1] 1373   15
```

```
# summary of transformed data
transformData %>% skimr::skim()
```

Table 4: Data summary

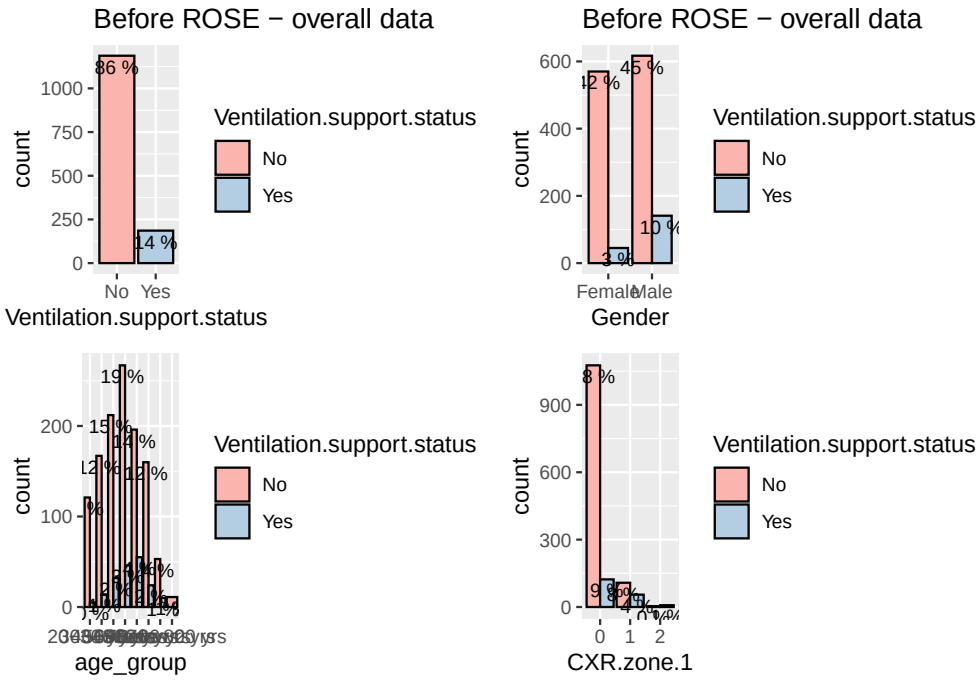
Name	Piped data
Number of rows	1373
Number of columns	15
Column type frequency:	
factor	15
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Gender	0	1	FALSE	2	Mal: 758, Fem: 615
Ventilation.support.status	0	1	FALSE	2	No: 1187, Yes: 186
CXR.zone.1	0	1	FALSE	3	0: 1199, 1: 163, 2: 11
CXR.zone.2	0	1	FALSE	3	0: 1269, 1: 97, 2: 7
CXR.zone.3	0	1	FALSE	3	0: 952, 1: 390, 2: 31
CXR.zone.4	0	1	FALSE	3	0: 1180, 1: 177, 2: 16
CXR.zone.5	0	1	FALSE	3	0: 839, 1: 481, 2: 53
CXR.zone.6	0	1	FALSE	3	0: 963, 1: 372, 2: 38
CXR.zone.7	0	1	FALSE	3	0: 1326, 1: 46, 2: 1
CXR.zone.8	0	1	FALSE	3	0: 1233, 1: 136, 2: 4
CXR.zone.9	0	1	FALSE	3	0: 1181, 1: 183, 2: 9
CXR.zone.10	0	1	FALSE	3	0: 950, 1: 391, 2: 32
CXR.zone.11	0	1	FALSE	3	0: 981, 1: 324, 2: 68
CXR.zone.12	0	1	FALSE	3	0: 809, 1: 491, 2: 73
age_group	0	1	FALSE	8	50-: 313, 60-: 251, 40-: 244, 70-: 184

```
# categorical features overall data
gg_ven<- transformData %>%
  ggplot(aes(x=`Ventilation.support.status`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  ggtitle("Before ROSE - overall data") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_gen <- transformData %>%
  ggplot(aes(x=`Gender`, fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  ggtitle("Before ROSE - overall data") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
```

```
  scale_fill_brewer(palette = "Pastel1")
gg_age <- transformData %>%
  ggplot(aes(x=`age_group`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr1 <- transformData %>%
  ggplot(aes(x=`CXR.zone.1`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
grid.arrange(gg_ven, gg_gen, gg_age, gg_cxr1, ncol=2)
```




```
gg_cxr2 <- transformData %>%
  ggplot(aes(x=`CXR.zone.2`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr3 <- transformData %>%
  ggplot(aes(x=`CXR.zone.3`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr4 <- transformData %>%
  ggplot(aes(x=`CXR.zone.4`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
```

```
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr5 <- transformData %>%
  ggplot(aes(x=`CXR.zone.5`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
#grid.arrange(gg_cxr2, gg_cxr3, gg_cxr4, gg_cxr5, ncol=2)

gg_cxr6 <- transformData %>%
  ggplot(aes(x=`CXR.zone.6`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr7 <- transformData %>%
  ggplot(aes(x=`CXR.zone.7`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr8 <- transformData %>%
  ggplot(aes(x=`CXR.zone.8`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr9 <- transformData %>%
  ggplot(aes(x=`CXR.zone.9`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
#grid.arrange(gg_cxr6, gg_cxr7, gg_cxr8, gg_cxr9, ncol=2)

gg_cxr10 <- transformData %>%
  ggplot(aes(x=`CXR.zone.10`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr11 <- transformData %>%
  ggplot(aes(x=`CXR.zone.11`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr12 <- transformData %>%
  ggplot(aes(x=`CXR.zone.12`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
```

```
    stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
#grid.arrange(gg_cxr10, gg_cxr11, gg_cxr12, ncol=2)
```

```
# split the dataset into two, 80% to Train and 20% to Test
set.seed(5)
idxSplit <- createDataPartition(transformData$Ventilation.support.status, p = 0.8, list=FALSE)
DstTrainModel <- transformData[idxSplit,] # 80% train data
DstTestModel <- transformData[-idxSplit,] # 20% test data

# summary of train data
DstTrainModel %>% skimr::skim()
```

Table 6: Data summary

Name	Piped data
Number of rows	1099
Number of columns	15
Column type frequency:	
factor	15
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Gender	0	1	FALSE	2	Mal: 605, Fem: 494
Ventilation.support.status	0	1	FALSE	2	No: 950, Yes: 149
CXR.zone.1	0	1	FALSE	3	0: 961, 1: 132, 2: 6
CXR.zone.2	0	1	FALSE	3	0: 1021, 1: 71, 2: 7
CXR.zone.3	0	1	FALSE	3	0: 768, 1: 307, 2: 24
CXR.zone.4	0	1	FALSE	3	0: 947, 1: 138, 2: 14
CXR.zone.5	0	1	FALSE	3	0: 679, 1: 379, 2: 41
CXR.zone.6	0	1	FALSE	3	0: 780, 1: 287, 2: 32
CXR.zone.7	0	1	FALSE	3	0: 1063, 1: 35, 2: 1
CXR.zone.8	0	1	FALSE	3	0: 988, 1: 107, 2: 4
CXR.zone.9	0	1	FALSE	3	0: 949, 1: 144, 2: 6
CXR.zone.10	0	1	FALSE	3	0: 766, 1: 308, 2: 25
CXR.zone.11	0	1	FALSE	3	0: 792, 1: 251, 2: 56
CXR.zone.12	0	1	FALSE	3	0: 655, 1: 388, 2: 56
age_group	0	1	FALSE	8	50-: 251, 60-: 196, 40-: 187, 30-: 150

```
# summary of test data
DstTestModel %>% skimr::skim()
```

Table 8: Data summary

Name	Piped data
Number of rows	274
Number of columns	15
Column type frequency:	
factor	15
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Gender	0	1	FALSE	2	Mal: 153, Fem: 121
Ventilation.support.status	0	1	FALSE	2	No: 237, Yes: 37
CXR.zone.1	0	1	FALSE	3	0: 238, 1: 31, 2: 5
CXR.zone.2	0	1	FALSE	2	0: 248, 1: 26, 2: 0
CXR.zone.3	0	1	FALSE	3	0: 184, 1: 83, 2: 7
CXR.zone.4	0	1	FALSE	3	0: 233, 1: 39, 2: 2
CXR.zone.5	0	1	FALSE	3	0: 160, 1: 102, 2: 12
CXR.zone.6	0	1	FALSE	3	0: 183, 1: 85, 2: 6
CXR.zone.7	0	1	FALSE	2	0: 263, 1: 11, 2: 0
CXR.zone.8	0	1	FALSE	2	0: 245, 1: 29, 2: 0
CXR.zone.9	0	1	FALSE	3	0: 232, 1: 39, 2: 3
CXR.zone.10	0	1	FALSE	3	0: 184, 1: 83, 2: 7
CXR.zone.11	0	1	FALSE	3	0: 189, 1: 73, 2: 12
CXR.zone.12	0	1	FALSE	3	0: 154, 1: 103, 2: 17
age_group	0	1	FALSE	8	50-: 62, 40-: 57, 60-: 55, 70-: 38

```
# check the target distribution, it is unbalance
table(DstTrainModel$Ventilation.support.status)
```

```
##
##  No Yes
## 950 149
```

```
# balance the target using ROSE (Random Over-Sampling)
set.seed(7)
rose_train <- ROSE(Ventilation.support.status ~ ., data  = DstTrainModel)$data
DstTrainModel <- rose_train

# check the target distribution, it is now balance
table(DstTrainModel$Ventilation.support.status)
```

```
##
##  No Yes
## 568 531
```

```
# summary of balanced train data
DstTrainModel %>% skimr::skim()
```

Table 10: Data summary

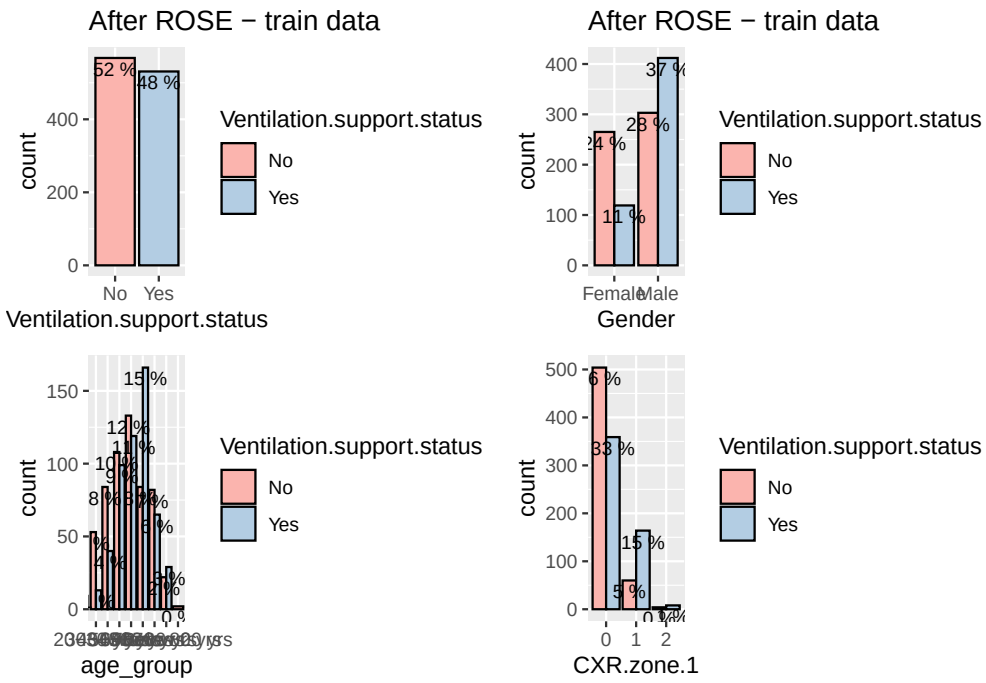
Name	Piped data
Number of rows	1099
Number of columns	15
Column type frequency:	
factor	15
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Gender	0	1	FALSE	2	Mal: 715, Fem: 384
Ventilation.support.status	0	1	FALSE	2	No: 568, Yes: 531
CXR.zone.1	0	1	FALSE	3	0: 863, 1: 224, 2: 12
CXR.zone.2	0	1	FALSE	3	0: 959, 1: 135, 2: 5
CXR.zone.3	0	1	FALSE	3	0: 581, 1: 470, 2: 48
CXR.zone.4	0	1	FALSE	3	0: 829, 1: 248, 2: 22
CXR.zone.5	0	1	FALSE	3	1: 552, 0: 494, 2: 53
CXR.zone.6	0	1	FALSE	3	0: 606, 1: 441, 2: 52
CXR.zone.7	0	1	FALSE	2	0: 1026, 1: 73, 2: 0
CXR.zone.8	0	1	FALSE	3	0: 937, 1: 156, 2: 6
CXR.zone.9	0	1	FALSE	3	0: 805, 1: 276, 2: 18
CXR.zone.10	0	1	FALSE	3	0: 594, 1: 460, 2: 45
CXR.zone.11	0	1	FALSE	3	0: 601, 1: 403, 2: 95
CXR.zone.12	0	1	FALSE	3	1: 552, 0: 457, 2: 90
age_group	0	1	FALSE	8	50-: 252, 60-: 250, 40-: 207, 70-: 147

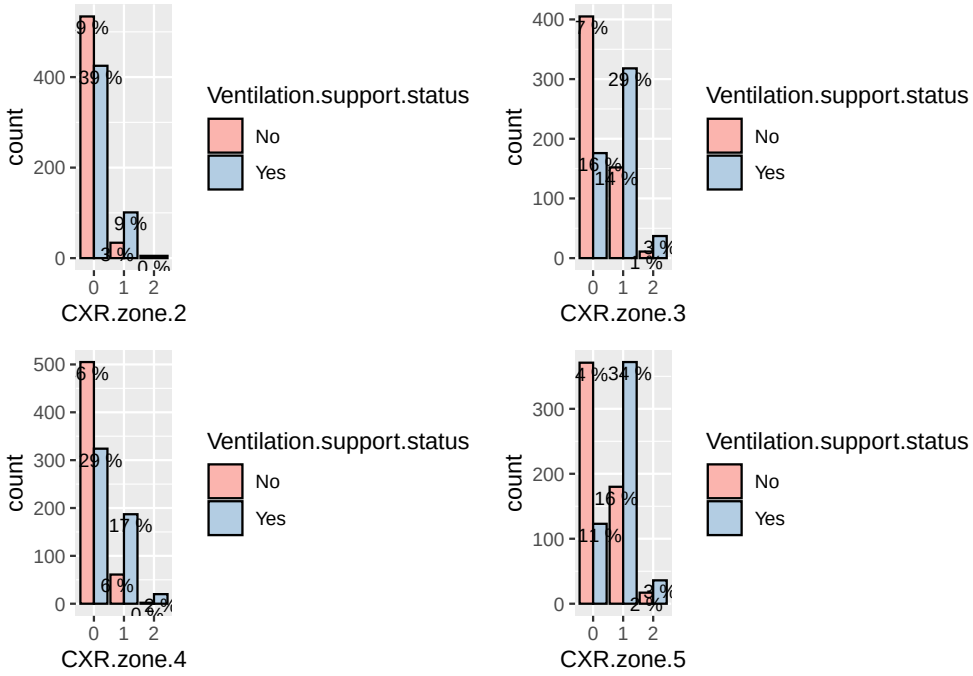
```
# categorical features train data
gg_vent<- DstTrainModel %>%
  ggplot(aes(x=`Ventilation.support.status`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  ggtitle("After ROSE - train data") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_gent <- DstTrainModel %>%
  ggplot(aes(x=`Gender`, fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  ggtitle("After ROSE - train data") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_aget <- DstTrainModel %>%
```

```
  ggplot(aes(x=`age_group`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr1t <- DstTrainModel %>%
  ggplot(aes(x=`CXR.zone.1`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
grid.arrange(gg_vent, gg_gent, gg_aget, gg_cxr1t, ncol=2)
```



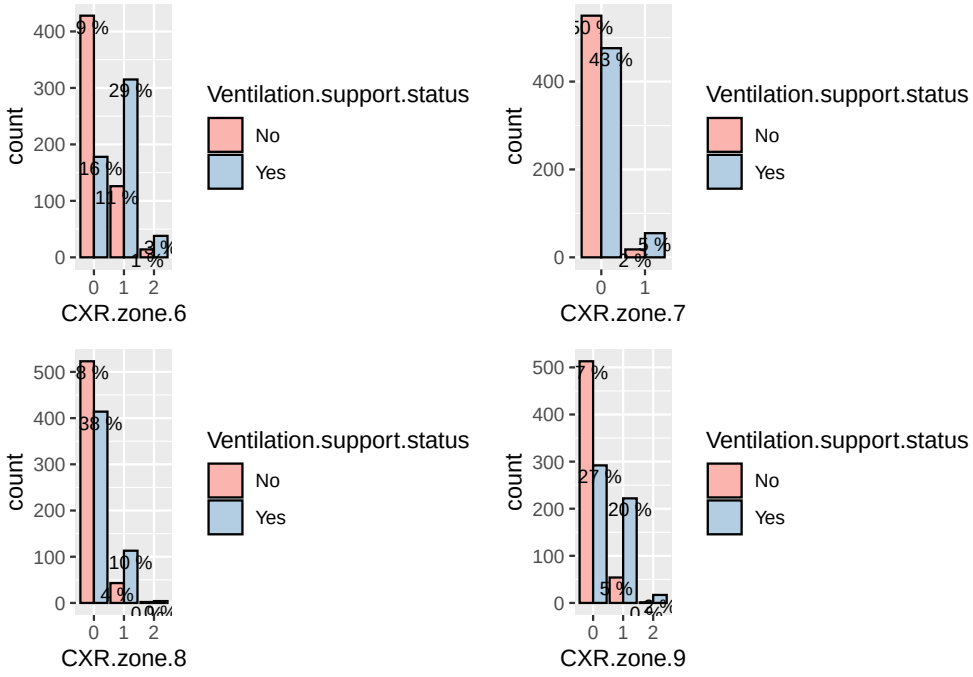

```
gg_cxr2t <- DstTrainModel %>%
  ggplot(aes(x=`CXR.zone.2`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr3t <- DstTrainModel %>%
  ggplot(aes(x=`CXR.zone.3`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr4t <- DstTrainModel %>%
  ggplot(aes(x=`CXR.zone.4`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
```

```
  scale_fill_brewer(palette = "Pastel1")
gg_cxr5t <- DstTrainModel %>%
  ggplot(aes(x=`CXR.zone.5`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
grid.arrange(gg_cxr2t, gg_cxr3t, gg_cxr4t, gg_cxr5t, ncol=2)
```

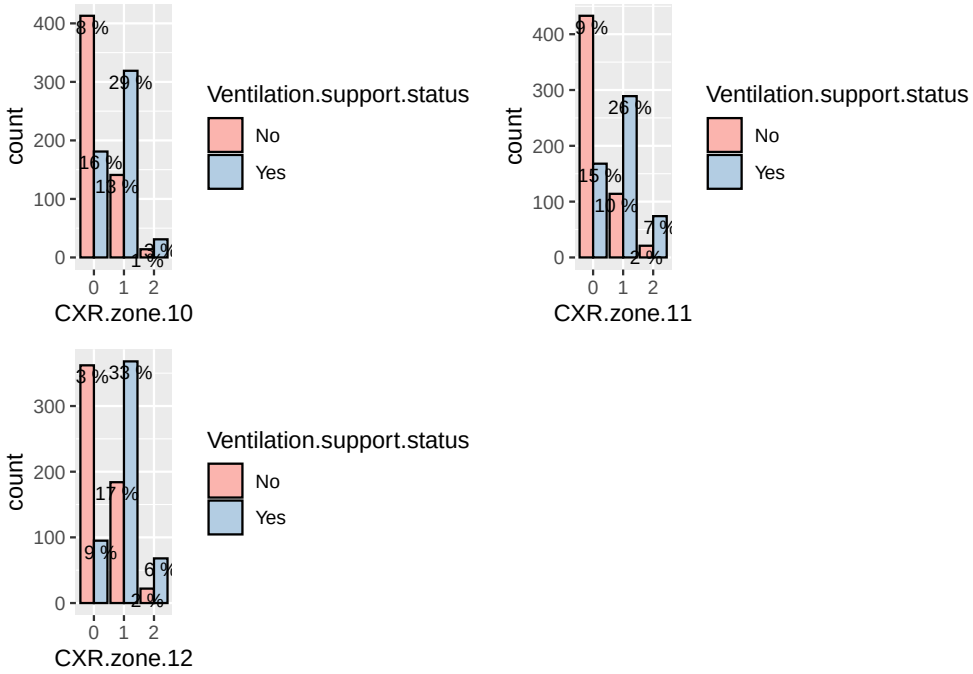



```
gg_cxr6t <- DstTrainModel %>%
  ggplot(aes(x=`CXR.zone.6`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr7t <- DstTrainModel %>%
  ggplot(aes(x=`CXR.zone.7`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr8t <- DstTrainModel %>%
  ggplot(aes(x=`CXR.zone.8`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr9t <- DstTrainModel %>%
  ggplot(aes(x=`CXR.zone.9`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
```

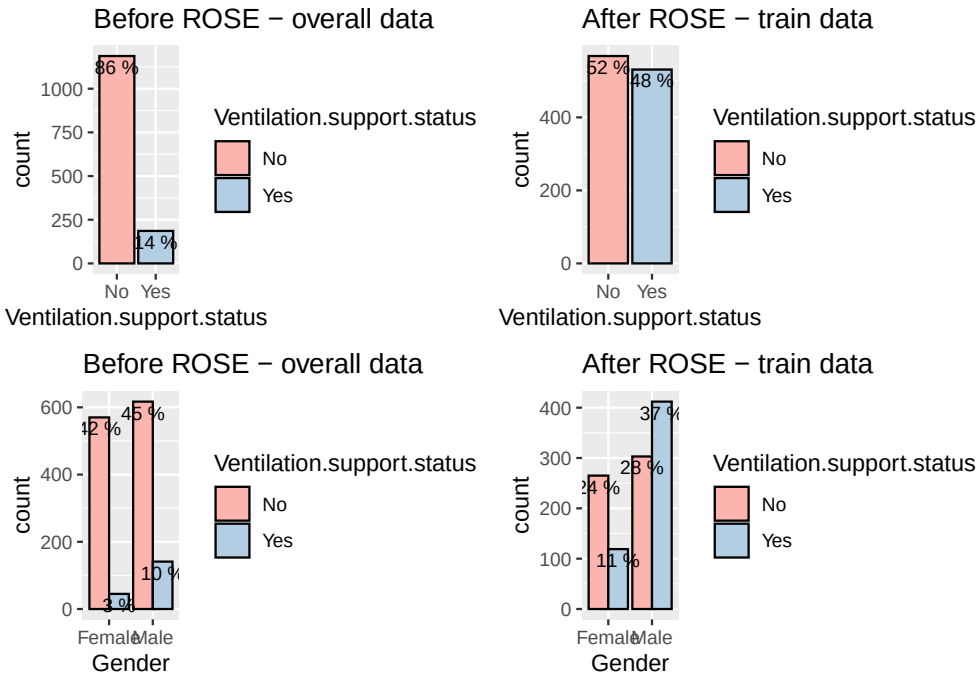
```
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
grid.arrange(gg_cxr6t, gg_cxr7t, gg_cxr8t, gg_cxr9t, ncol=2)
```




```
gg_cxr10t <- DstTrainModel %>%
  ggplot(aes(x=`CXR.zone.10`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr11t <- DstTrainModel %>%
  ggplot(aes(x=`CXR.zone.11`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
gg_cxr12t <- DstTrainModel %>%
  ggplot(aes(x=`CXR.zone.12`,fill=Ventilation.support.status)) +
  geom_bar(position='dodge',colour="black") +
  geom_text(aes(label=paste(round(after_stat(count*100/sum(count))),"%")),
     stat='count', vjust = 1.3, position=position_dodge(width=0.9),size=3) +
  scale_fill_brewer(palette = "Pastel1")
grid.arrange(gg_cxr10t, gg_cxr11t, gg_cxr12t, ncol=2)
```




```
grid.arrange(gg_ven, gg_vent, gg_gen, gg_gent, ncol=2)
```



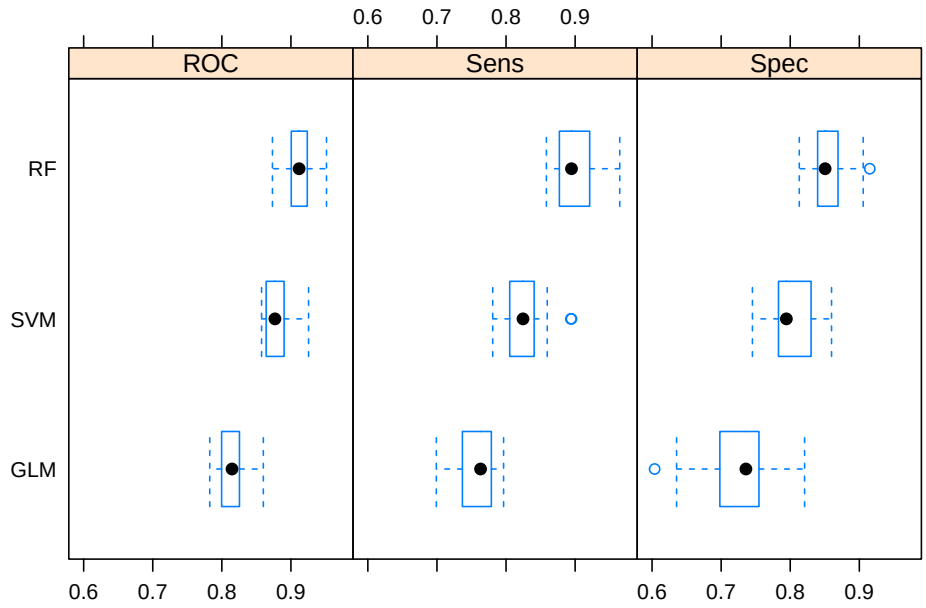

```
# set method to 5-fold repeated cross validation
control <- trainControl(
  method = "repeatedcv",
  number = 5,
  repeats = 5,
  classProbs = TRUE,
```

```
  summaryFunction = twoClassSummary)

# train model using Generalized Linear Model (GLM)
#set.seed(7)
#fit.glm <- train(Ventilation.support.status ~ ., data=DstTrainModel, method="glm", trControl=control)
load("fitglm.Rdata")

# train model using Support Vector Machine (SVM)
#set.seed(7)
#fit.svmr <- train(Ventilation.support.status ~ ., data=DstTrainModel, method="svmRadial", trControl=co
#load("fitsvm.Rdata")
load("fitsvmr.Rdata")

# train model using Random Forest (RF)
#set.seed(7)
#fit.rf <- train(Ventilation.support.status ~ ., data=DstTrainModel, method="rf", trControl=control)
load("fitrf.Rdata")
```

```
# summarize accuracy of models
results <- resamples(list(
    'GLM'=fit.glm,
    'SVM'=fit.svmr,
    'RF'=fit.rf
))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: GLM, SVM, RF
## Number of resamples: 25
##
## ROC
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM 0.7825053 0.7996524 0.8147237 0.8168971 0.8255134 0.8600768    0
## SVM 0.8574975 0.8641259 0.8768651 0.8798808 0.8901026 0.9255301    0
## RF  0.8732622 0.9004882 0.9118298 0.9113544 0.9236594 0.9514944    0
##
## Sens
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM 0.6991150 0.7368421 0.7631579 0.7559975 0.7787611 0.7964602    0
## SVM 0.7807018 0.8053097 0.8245614 0.8260550 0.8407080 0.8947368    0
## RF  0.8584071 0.8771930 0.8947368 0.9010526 0.9210526 0.9646018    0
##
## Spec
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM 0.6037736 0.6981132 0.7358491 0.7258826 0.7547170 0.8207547    0
## SVM 0.7452830 0.7830189 0.7943925 0.8003809 0.8301887 0.8598131    0
## RF  0.8130841 0.8396226 0.8504673 0.8576265 0.8691589 0.9150943    0
```

```
# compare ROC of models
bwplot(results, layout=c(3,1))
```




```
# summary of GLM model, ROC used to select optimize model
fit.glm
```

```
## Generalized Linear Model
##
## 1099 samples
##   14 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 879, 879, 879, 880, 879, 880, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.8168971  0.7559975  0.7258826
```

```
# average accuracy of GLM during training
confusionMatrix(fit.glm)
```

```
## Cross-Validated (5 fold, repeated 5 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   No  Yes
```

```
##        No  39.1 13.2
##        Yes 12.6 35.1
##
##  Accuracy (average) : 0.7414
```

```
# summary of SVM model, ROC used to select optimize model
fit.svmr
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 1099 samples
##   14 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 879, 879, 879, 880, 879, 880, ...
## Resampling results across tuning parameters:
##
##   C     ROC        Sens       Spec
##   0.25  0.8541595  0.7281633  0.8038159
##   0.50  0.8652230  0.7739419  0.7861012
##   1.00  0.8798808  0.8260550  0.8003809
##
## Tuning parameter 'sigma' was held constant at a value of 0.2083333
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.2083333 and C = 1.
```

```
# average accuracy of SVM during training
confusionMatrix(fit.svmr)
```

```
## Cross-Validated (5 fold, repeated 5 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   No  Yes
##        No  42.7  9.6
##        Yes  9.0 38.7
##
##  Accuracy (average) : 0.8136
```

```
# summary of RF model, ROC used to select optimize model
fit.rf
```

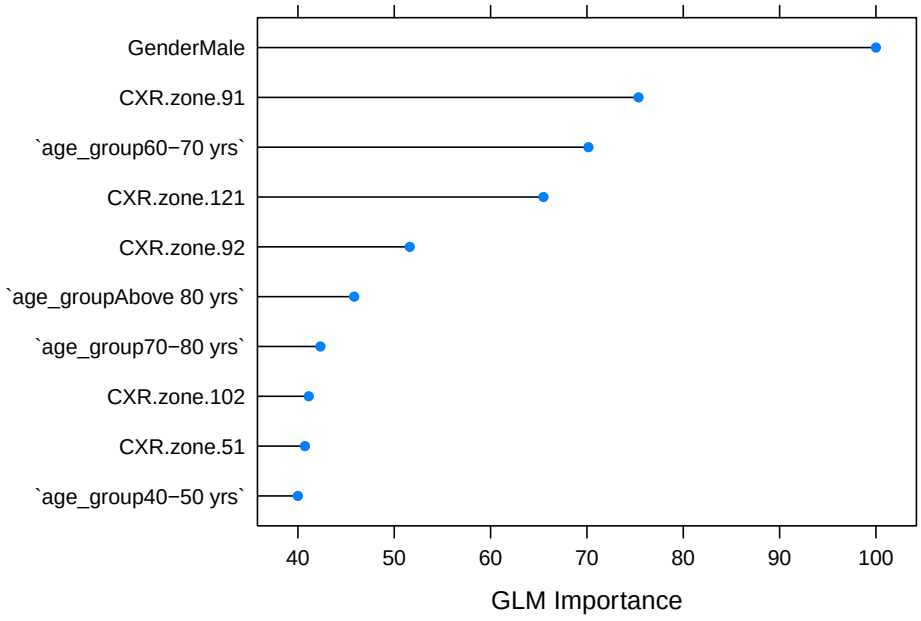
```
## Random Forest
##
## 1099 samples
##   14 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
```

```
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 879, 879, 879, 880, 879, 880, ...
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##    2    0.8472023  0.7232542  0.8278998
##   17    0.9106864  0.8954262  0.8602680
##   32    0.9113544  0.9010526  0.8576265
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 32.
```

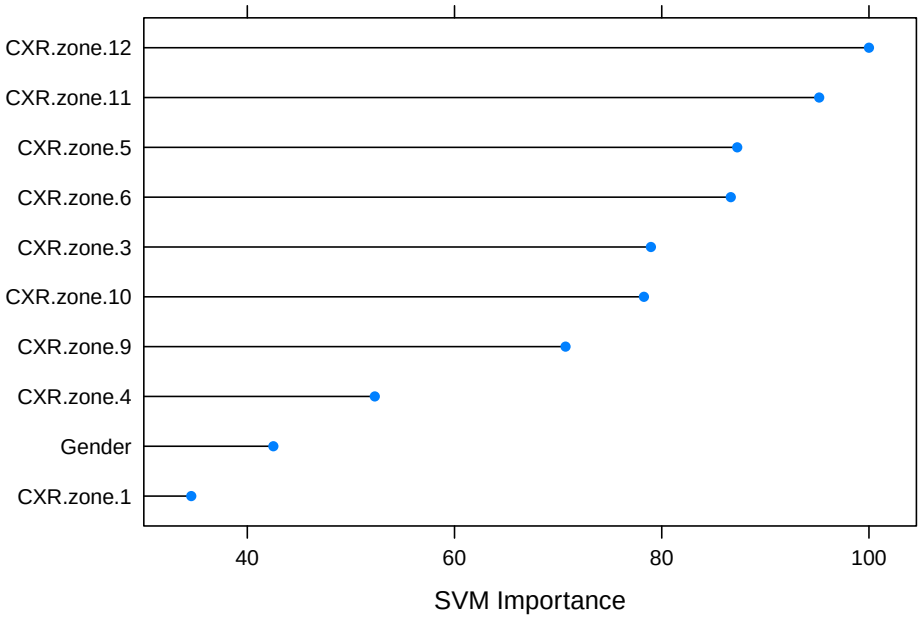
```
# average accuracy of RF during training
confusionMatrix(fit.rf)
```

```
## Cross-Validated (5 fold, repeated 5 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   No  Yes
##        No  46.6  6.9
##        Yes  5.1 41.4
##
##  Accuracy (average) : 0.8801
```

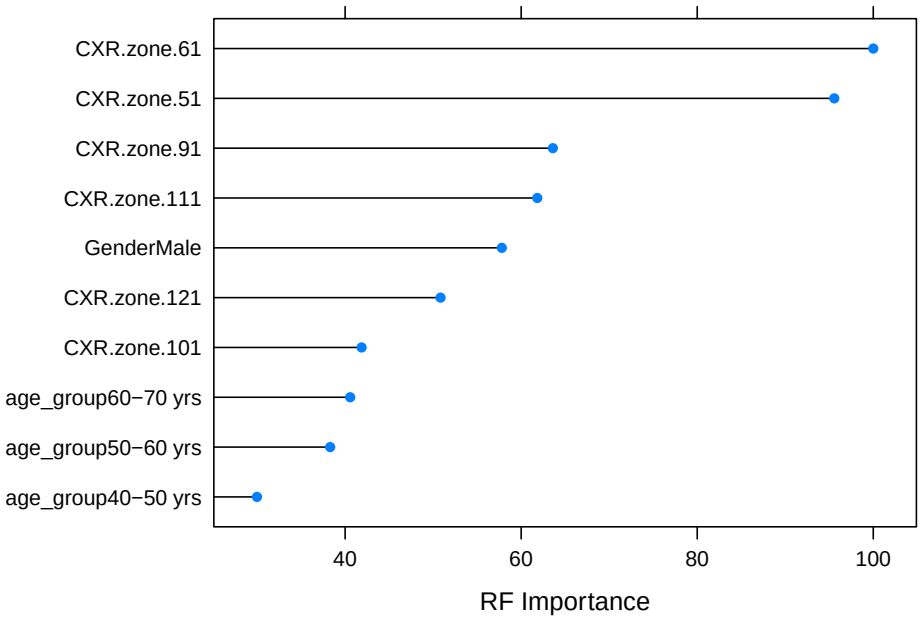
```
# variable importance plot for each model
#v_glm <- plot(varImp(fit.glm, scale=TRUE), top=10, xlab="GLM Importance")
#v_svmr <- plot(varImp(fit.svmr, scale=TRUE), top=10, , xlab="SVM Importance")
#v_rf <- plot(varImp(fit.rf, scale=TRUE), top=10, , xlab="RF Importance")
#grid.arrange(v_glm, v_svmr, v_rf, ncol=2)
plot(varImp(fit.glm, scale=TRUE), top=10, xlab="GLM Importance")
```




```
plot(varImp(fit.svmr, scale=TRUE), top=10, , xlab="SVM Importance")
```



```
plot(varImp(fit.rf, scale=TRUE), top=10, , xlab="RF Importance")
```




```
# function to test the prediction model using test dataset
pred_func <- function(TestFit) {
  preds_bag <- bind_cols(
    predict(TestFit, newdata = DstTestModel, type = "prob"),
    Predicted = predict(TestFit, newdata = DstTestModel, type = "raw"),
    Actual = DstTestModel$Ventilation.support.status
    )
  return(preds_bag)
}

# predict test dataset using GLM model
glm_preds <- pred_func(fit.glm)
glm_cm <- confusionMatrix(glm_preds$Predicted, reference = glm_preds$Actual, positive = "No")
# summary of test dataset prediction using GLM model
glm_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  177  12
##        Yes  60  25
##
##                Accuracy : 0.7372
##                  95% CI : (0.6809, 0.7883)
##     No Information Rate : 0.865
##     P-Value [Acc > NIR] : 1
##
```

```
##                   Kappa : 0.273
##
##  Mcnemar's Test P-Value : 3.042e-08
##
##             Sensitivity : 0.7468
##             Specificity : 0.6757
##          Pos Pred Value : 0.9365
##          Neg Pred Value : 0.2941
##              Prevalence : 0.8650
##          Detection Rate : 0.6460
##    Detection Prevalence : 0.6898
##       Balanced Accuracy : 0.7113
##
##        'Positive' Class : No
##
```

```
# predict test dataset using SVM model
svmr_preds <- pred_func(fit.svmr)
svmr_cm <- confusionMatrix(svmr_preds$Predicted, reference = svmr_preds$Actual, positive = "No")
# summary of test dataset prediction using SVM model
svmr_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  187  13
##        Yes  50  24
##
##                Accuracy : 0.7701
##                  95% CI : (0.7156, 0.8186)
##     No Information Rate : 0.865
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3078
##
##  Mcnemar's Test P-Value : 5.745e-06
##
##             Sensitivity : 0.7890
##             Specificity : 0.6486
##          Pos Pred Value : 0.9350
##          Neg Pred Value : 0.3243
##              Prevalence : 0.8650
##          Detection Rate : 0.6825
##    Detection Prevalence : 0.7299
##       Balanced Accuracy : 0.7188
##
##        'Positive' Class : No
##
```

```
# predict test dataset using RF model
rf_preds <- pred_func(fit.rf)
rf_cm <- confusionMatrix(rf_preds$Predicted, reference = rf_preds$Actual, positive = "No")
```

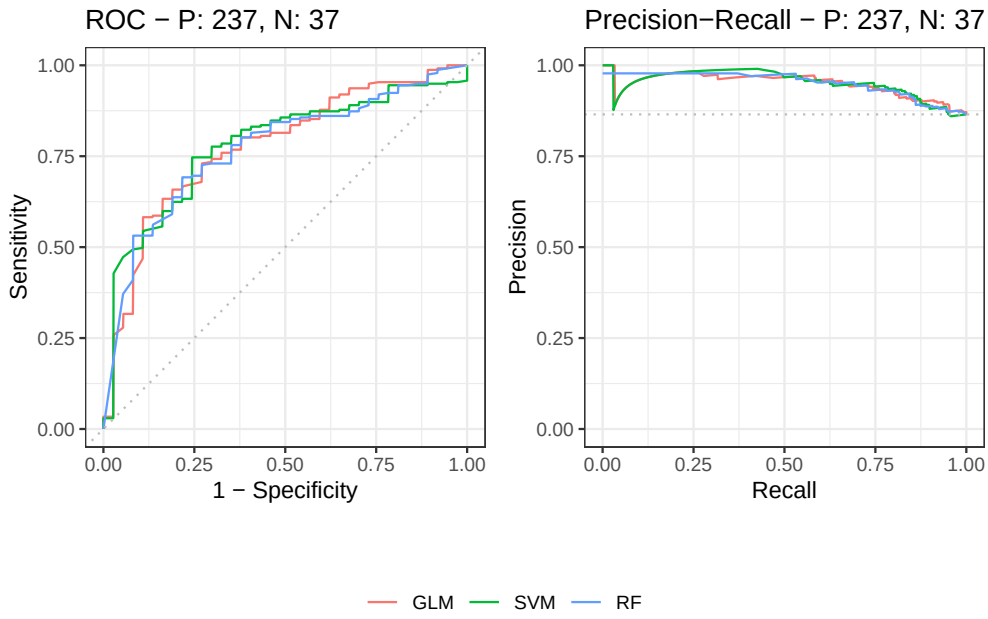
```r
# summary of test dataset prediction using RF model
rf_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  200  17
##        Yes  37  20
##
##                Accuracy : 0.8029
##                  95% CI : (0.7508, 0.8483)
##     No Information Rate : 0.865
##     P-Value [Acc > NIR] : 0.998368
##
##                   Kappa : 0.313
##
##  Mcnemar's Test P-Value : 0.009722
##
##             Sensitivity : 0.8439
##             Specificity : 0.5405
##          Pos Pred Value : 0.9217
##          Neg Pred Value : 0.3509
##              Prevalence : 0.8650
##          Detection Rate : 0.7299
##    Detection Prevalence : 0.7920
##       Balanced Accuracy : 0.6922
##
##        'Positive' Class : No
##
```

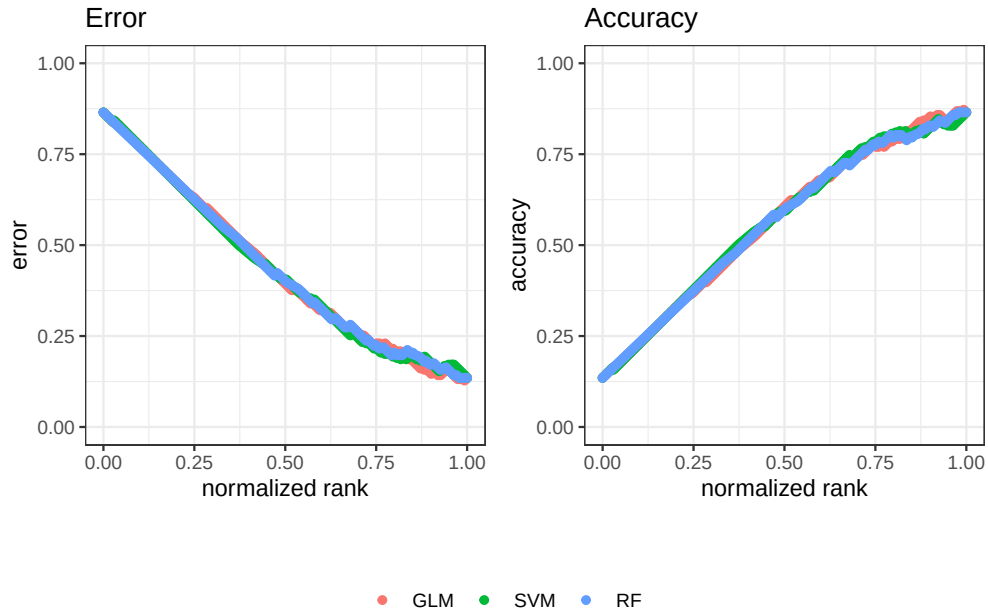
```r
# function to take out performance metrics from ML model using test dataset
measure <- function(cm,preds,name){
  meas <- data.frame(matrix(ncol=0,nrow=1))
  meas$model <- name
  meas$Accuracy <- cm$overall['Accuracy']
  meas$Sensitivity <- cm$byClass['Sensitivity']
  meas$Specificity <- cm$byClass['Specificity']
  meas$Precision <- cm$byClass['Precision']
  meas$Recall <- cm$byClass['Recall']
  meas$F1 <- cm$byClass['F1']
  meas$auc <- Metrics::auc(actual = preds$Actual == "No", preds$No)
  return(meas)
}

# combine the results in measAll
measAll <- measure(glm_cm,glm_preds,"glm")
measAll <- rbind(measAll, measure(svmr_cm,svmr_preds,"svmr"))
measAll <- rbind(measAll, measure(rf_cm,rf_preds,"rf"))
rownames(measAll) <- c()
# show the performance metric results in table form
arrange(measAll,desc(Accuracy))
```

```
##   model  Accuracy Sensitivity Specificity Precision    Recall        F1
## 1    rf 0.8029197   0.8438819   0.5405405 0.9216590 0.8438819 0.8810573
## 2  svmr 0.7700730   0.7890295   0.6486486 0.9350000 0.7890295 0.8558352
## 3   glm 0.7372263   0.7468354   0.6756757 0.9365079 0.7468354 0.8309859
##         auc
## 1 0.7718098
## 2 0.7803626
## 3 0.7763713
```

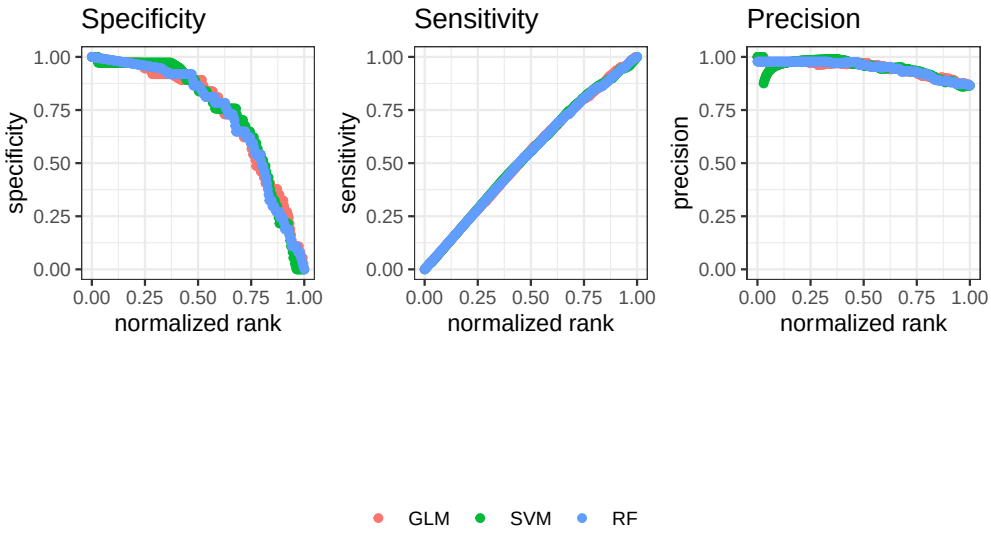
```r
# take predicted result from multiple models and converts them to a list
scores_list <- join_scores(
  predict(fit.glm, newdata = DstTestModel, type = "prob")$No,
  predict(fit.svmr, newdata = DstTestModel, type = "prob")$No,
  predict(fit.rf, newdata = DstTestModel, type = "prob")$No
)

# take actual result and converts them to a list
labels_list <- join_labels(
  DstTestModel$Ventilation.support.status,
  DstTestModel$Ventilation.support.status,
  DstTestModel$Ventilation.support.status
)

# calculates ROC and precision-recall curves
metric1 <- evalmod(
  scores = scores_list,
  labels = labels_list,
  modnames = c("GLM","SVM","RF"),
  posclass = "No")

# calculates other performance metrics using
metric2 <- evalmod(
  scores = scores_list,
  labels = labels_list,
  modnames = c("GLM","SVM","RF"),
  mode = "basic",
  posclass = "No")

# show ROC and precision-recall curves
autoplot(metric1)
```




```
# show error and accuracy curves
autoplot(metric2, c("error", "accuracy"))
```




```
# show specificity, sensitivity and precision curves
autoplot(metric2, c("specificity", "sensitivity", "precision"))
```




```
# summary of AUC and ROC
metric1
```

```
##
##     === AUCs ===
##
##      Model name Dataset ID Curve type       AUC
##    1        GLM          1        ROC 0.7763713
##    2        GLM          1        PRC 0.9495323
##    3        SVM          1        ROC 0.7803626
##    4        SVM          1        PRC 0.9523395
##    5         RF          1        ROC 0.7718098
##    6         RF          1        PRC 0.9520433
##
##
##     === Input data ===
##
##      Model name Dataset ID # of negatives # of positives
##    1        GLM          1             37            237
##    2        SVM          1             37            237
##    3         RF          1             37            237
```