Mortality Prediction in the ICU

BIOS 685 Final Project

Anthony Vicenti, Chen Chen, Meng Lu

12/11/2019

Contents

Abstract

Health care is one of the most exciting frontiers in data mining and machine learning. Successful adoption of electronic health records (EHRs) created an explosion in digital clinical data available for analysis, but progress in machine learning for healthcare research has been difficult to measure because of the absence of publicly available benchmark data sets (Harutyunyan et al. 2017). To address this problem, we proposed four clinical prediction benchmarks using data derived from the publicly available Medical Information Mart for Intensive Care (MIMIC-III) database. Our models show strong predition power (e.g., 0.89) for motarlity prediction across 12605 patients we extracted.

Introduction

Improved mortality prediction for patients in intensive care units (ICU) remains an important challenge. Many severity scores have been proposed but validation studies have concluded that they are not adequately calibrated. Many flexible algorithms are available, yet none of these individually outperform all others regardless of context. The aim of the present study was to compare several mortality prediction algorithms for ICU patients using curated covariates from MIMIC-III database(Silva et al. 2012). The rest of the paper is organized as follows: Methods, Results, Discussions, Code Appendix, and References.

Methods

Study Design and Study Size

The MIMIC-III database is a large database that contains health-related data for over 40,000 patients who stayed in ICU unites between 2001 and 2012 at the Beth Israel Deaconess Medical Center. This data base includes variables such as demographics, vital signs, laboratory test, and mortality outcomes.

Our data set contained all individuals between 18 and 90 because individuals over 90 had their ages masked. We also only looked at those patients who reported an admission type of emergency. This is the most important case compared to the elective case. Since there were multiple ICU stays for some of the patients, we only included the first stay. Finally, we did a complete case analysis and ended up with 12605 individuals.

Measures/Variables

All consecutive patients were included in the MIMIC-III database. The data acquisition process was not visible to staff and did not interfere with the clinical care of patients or methods of monitoring. Only patients with a single ICU admission per hospital stay were considered for the present analysis. Two categories of data were collected: clinical data, aggregated from ICU information systems and hospital archives, and high-resolution physiologic data (waveforms and time series of derived physiologic measurements), recorded on bedside monitors. Clinical data were obtained from the CareVue Clinical Information System deployed in all study ICUs, and from hospital electronic archives. The data included time-stamped nurse-verified physiologic measurements (e.g., hourly documentation of heart rate, arterial blood pressure, pulmonary artery pressure), nurses' and respiratory therapists' progress notes, continuous intravenous (IV) drip medications, fluid balances, patient demographics, interpretations of imaging studies, physician orders, discharge summaries, and ICD-9 codes. Comprehensive diagnostic laboratory results (e.g., blood chemistry, complete blood counts, arterial blood gases, microbiology results) were obtained from the patient's entire hospital stay including periods outside the ICU (Vincent et al. 2018).

Statistical Ananlysis

We firstly conducted some descriptive analysis on the data we extracted. As known to all, there are a lot of missing values in MIMIC data. So we proposed one missing imputation strategy to deal with the missing data. We will compare the results of before and after imputation. We then chose to use Random Forest (RF) (Sadeghi, Banerjee, and Romine 2018), Gradient Boosting Method (GBM) (Feng et al. 2018), Feed-Forward Neural Network (NN), and xgboost (Johnson and Mark 2017) to predict the mortality in the ICU. Random forest is one of the most popular decision tree model. It is an ensemble learning method for classification, regression. Gradient Boosting Method is a new method for classification and regresion based on the weak learner. It builds the model in a stage-wise fashion like other boosting methods do. The optimizer focus on the differentiable loss function. Feed-Forward Neural Network was the first and simplest of artificial network. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes and to the output nodes. xgboost stands for eXtreme Gradient Boosting, which belongs to a broader collection of tools under the umbrella of the distributed machine learning community. xgboost is the most popular machine learning method, especially, for the kaggle competition.

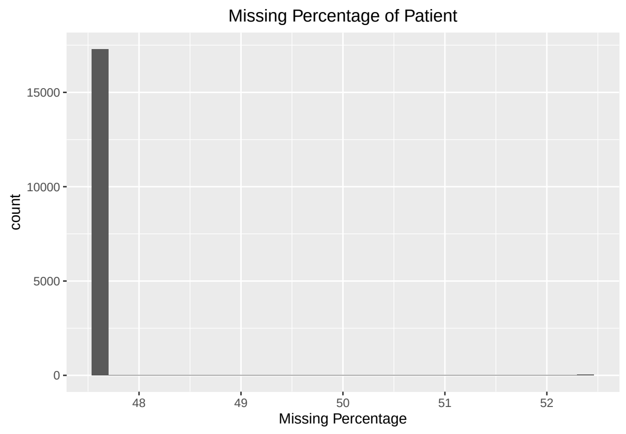
For all the machine learning models, we used h2o package from R version 3.6.1. h2o is a fully open source, distributed in-memory machine learning platform with linear scalability. h2o supports the most widely used statistical and machine learning algorithms including gradient boosted machines, generalized linear models, deep learning and more. h2o also has an industry leading AutoML functionality that automatically runs through all the algorithms and their hyperparameters to produce a leaderboard of the best models. More details about H2O, please follow the link to see the products, solutions about the H2O.
For installing H2O, please refer R and Python.

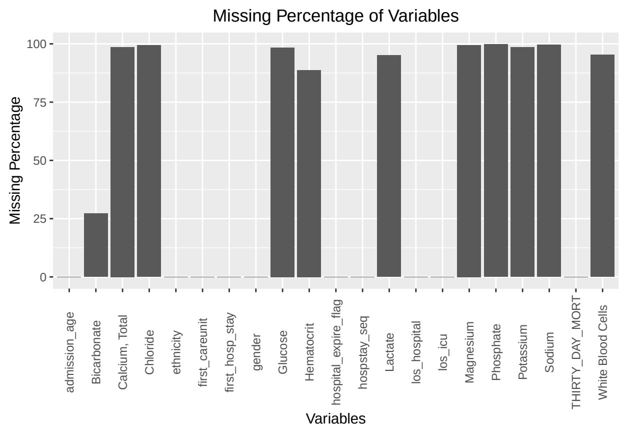
For missing value imputation, we applied missForest. We split the imputed data or complete data into three parts, trainging set, validation set and testing set.

Results

Descriptive Analysis

We first conducted a descriptive analysis to show missing values across variables and observations of the dataset we extracted. Also we want to know the summary statistics of all the variables.





As shown in Figures above, most of patients contain the missing values around 48%. And around half of the variables have missing around 100%. Thus, we decided to use the complete cases and also applied missing value imputation first.

Characteristic	**Statistic**	**0**, N = 10714	**1**, N = 6617	**p-value**[1]
first_careunit				<0.001
CCU	n / N (%)	1900 / 10714 (18%)	937 / 6617 (14%)	
CSRU	n / N (%)	1024 / 10714 (9.6%)	365 / 6617 (5.5%)	
MICU	n / N (%)	5345 / 10714 (50%)	3494 / 6617 (53%)	
SICU	n / N (%)	1594 / 10714 (15%)	1091 / 6617 (16%)	
TSICU	n / N (%)	851 / 10714 (7.9%)	730 / 6617 (11%)	
gender				0.8
F	n / N (%)	4713 / 10714 (44%)	2926 / 6617 (44%)	
M	n / N (%)	6001 / 10714 (56%)	3691 / 6617 (56%)	
los_hospital	mean (SD)	12 (12)	11 (13)	<0.001
admission_age	mean (SD)	68 (14)	69 (15)	<0.001
hospital_expire_flag				<0.001
0	n / N (%)	10714 / 10714 (100%)	1776 / 6617 (27%)	
1	n / N (%)	0 / 10714 (0%)	4841 / 6617 (73%)	
hospstay_seq	mean (SD)	1.82 (2.34)	1.49 (1.26)	<0.001
first_hosp_stay				<0.001
FALSE	n / N (%)	3445 / 10714 (32%)	1731 / 6617 (26%)	
TRUE	n / N (%)	7269 / 10714 (68%)	4886 / 6617 (74%)	
los_icu	mean (SD)	4.5 (6.8)	5.7 (7.7)	<0.001
Bicarbonate	mean (SD)	25.3 (5.1)	24.4 (5.4)	<0.001
Unknown	n	2766	1960	

[1]Statistical tests performed: chi-square test of independence; Wilcoxon rank-sum test

Above is all the summary statistics of the complete dataset.

Statistical Analysis

Complete Dataset

We firstly applied our analysis models on complete dataset. We used h2o to build up the training environment and split the data into training, validation, and testing sets randomly. Then We applied all four machine learning models and compare their validation performance and testing performance.

Validation Performance

Validation Results: Deep Neural Network (DNN)

	0	1	Error	Rate
0	4782	3	0.0006270	=3/4785
1	770	1995	0.2784810	=770/2765
Totals	5552	1998	0.1023841	=773/7550

Validation Results: Gradient Boosting Model (GBM)

	0	1	Error	Rate
0	4597	188	0.0392894	=188/4785
1	440	2325	0.1591320	=440/2765
Totals	5037	2513	0.0831788	=628/7550

Validation Results: Random Forest (RF)

	0	1	Error	Rate
0	4775	10	0.0020899	=10/4785
1	778	1987	0.2813743	=778/2765
Totals	5553	1997	0.1043709	=788/7550

Validation Results: xgboost

	0	1	Error	Rate
0	4785	0	0	=0/4785
1	0	2765	0	=0/2765
Totals	4785	2765	0	=0/7550

Above four tables shows the validation performance of the four models on complete case training set. It shows the four models are fitting pretty well with only around 0.1 error rate. Now we use AUC to quantify the general performance of each model.

Validation Results: AUC

AUC	Methods
0.8755919	RF
0.9635437	GBM
0.8973489	DNN
1.0000000	xgboost

Not surprisingly, xgboost seems to be overfitted on training dataset. Now we test their performance on the testing data.

Testing Performance

Testing Results: Deep Neural Network (DNN)

	0	1	Error	Rate
0	1541	0	0.0000000	=0/1541
1	275	664	0.2928647	=275/939
Totals	1816	664	0.1108871	=275/2480

Testing Results: Gradient Boosting Model (GBM)

	0	1	Error	Rate
0	1540	1	0.0006489	=1/1541
1	271	668	0.2886049	=271/939
Totals	1811	669	0.1096774	=272/2480

Testing Results: Random Forest (RF)

	0	1	Error	Rate
0	1540	1	0.0006489	=1/1541
1	272	667	0.2896699	=272/939
Totals	1812	668	0.1100806	=273/2480

Testing Results: xgboost

	0	1	Error	Rate
0	1540	1	0.0006489	=1/1541
1	275	664	0.2928647	=275/939
Totals	1815	665	0.1112903	=276/2480

Testing Results: AUC

AUC	Methods
0.8702981	RF
0.8724063	GBM
0.8741513	DNN
0.8707629	xgboost

The testing performance is surprisingly good for all these models because we did not do any parameter tuning for the reason of computational time. However, these experience-based parameters did a very good job. Then we checked the AUC as well, it seems all four models we chose have very similar performance, Random Forest is slightly better than the other three though.

Imputated Dataset

Again, we did the exactly same analysis on the imputated dataset.

Validation Performance

Validation Results (Imputation): Deep Neural Network (DNN)

	0	1	Error	Rate
0	6138	1	0.0001629	=1/6139
1	1069	2812	0.2754445	=1069/3881
Totals	7207	2813	0.1067864	=1070/10020

Validation Results (Imputation): Gradient Boosting Model (GBM)

	0	1	Error	Rate
0	6146	245	0.0383352	=245/6391
1	646	3389	0.1600991	=646/4035
Totals	6792	3634	0.0854594	=891/10426

Validation Results (Imputation): Random Forest (RF)

	0	1	Error	Rate
0	6381	10	0.0015647	=10/6391
1	1103	2932	0.2733581	=1103/4035
Totals	7484	2942	0.1067523	=1113/10426

As shown before, these tables are the validation performances of the four models on the imputated dataset. It is slightly better than complete case dataset.

Validation Results (Imputation): xgboost

	0	1	Error	Rate
0	6391	0	0	=0/6391
1	0	4035	0	=0/4035
Totals	6391	4035	0	=0/10426

Validation Results (Imputation): AUC

AUC	Methods
0.8799242	RF
0.9632160	GBM
0.8959253	DNN
1.0000000	xgboost

Testing Results (Imputation): Deep Neural Network (DNN)

	0	1	Error	Rate
0	2088	0	0.0000000	=0/2088
1	343	958	0.2636434	=343/1301
Totals	2431	958	0.1012098	=343/3389

Testing Results (Imputation): Gradient Boosting Model (GBM)

	0	1	Error	Rate
0	2087	1	0.0004789	=1/2088
1	343	958	0.2636434	=343/1301
Totals	2430	959	0.1015049	=344/3389

Testing Results (Imputation): Random Forest (RF)

	0	1	Error	Rate
0	2086	2	0.0009579	=2/2088
1	340	961	0.2613374	=340/1301
Totals	2426	963	0.1009147	=342/3389

Testing Results (Imputation): xgboost

	0	1	Error	Rate
0	2088	0	0.0000000	=0/2088
1	341	960	0.2621061	=341/1301
Totals	2429	960	0.1006197	=341/3389

Testing Results (Imputation): AUC

AUC	Methods
0.8949462	RF
0.8935755	GBM
0.8960647	DNN
0.8901077	xgboost

Testing Performance

The testing performance is consistent with the validation performance - they are both slightly better on imputated dataset. This makes sense because we added one more variable to the model. However, no matter in training or testing datasets, the performance of all four models are extremely good. This is mainly because we have over 10,000 observations but only 10 predictors. Our future direction will be to extract more variables to construct a better prediction model.

Discussion

The AUC values for all methods were close to the same value, 0.872, 0.868, 0.867, 0.868 for the random forest, gradient boosting method, neural network, and xgboost respectively. The random forest method produced slightly better results than the rest. After we used imputation, the results were all better for all the models. The AUC values rose to 0.891, 0.893, 0.890, 0.890 for the random forest, gradient boosting method, neural network, and xgboost respectively. After imputation, the gradient boosting method produced the best results, but only by 0.002.
Although our models produced high AUC values, there were still more predictor variables that could have been added into the models. This includes the vitals, such as blood pressure and heart rate, and the variables that produced the different severity scores. These variables could have increased our AUC even higher.
Hospitals that are attempting to predict mortality outcomes should use the gradient boosting method on imputed data to receive the best results. This can lead to more efficient ICUs that can target the most vulnerable patients and reduce overall death.

Appendix

Data Extraction Code

```
# Load configuration settings
dbdriver <- 'PostgreSQL'
host  <- '127.0.0.1'
port  <- '5432'
user  <- 'postgres'
password <- 'postgres'
dbname <- 'mimic'
schema <- 'mimiciii'

# Connect to the database using the configuration settings
con <- dbConnect(dbDriver(dbdriver),
                 dbname = dbname,
                 user = user,
                 password = password)

# Set the default schema
dbExecute(con, paste("SET search_path TO ", schema, sep=" "))

icustay <- tbl(con,"icustays") %>% select(1-10)
icustay_detail <- dbGetQuery(con,'
SELECT ie.subject_id, ie.hadm_id, ie.icustay_id, ie.first_careunit

-- patient level factors
, pat.gender, pat.dod

-- hospital level factors
, adm.admittime, adm.dischtime
, ROUND( (CAST(EXTRACT(epoch FROM adm.dischtime - adm.admittime)/(60*60*24) AS numeric)), 4) AS los_hosp
, ROUND( (CAST(EXTRACT(epoch FROM adm.admittime - pat.dob)/(60*60*24*365.242) AS numeric)), 4) AS admiss
, adm.ethnicity, adm.admission_type
, adm.hospital_expire_flag
, DENSE_RANK() OVER (PARTITION BY adm.subject_id ORDER BY adm.admittime) AS hospstay_seq
, CASE
    WHEN DENSE_RANK() OVER (PARTITION BY adm.subject_id ORDER BY adm.admittime) = 1 THEN True
    ELSE False END AS first_hosp_stay

-- icu level factors
, ie.intime, ie.outtime
, ROUND( (CAST(EXTRACT(epoch FROM ie.outtime - ie.intime)/(60*60*24) AS numeric)), 4) AS los_icu
, DENSE_RANK() OVER (PARTITION BY ie.hadm_id ORDER BY ie.intime) AS icustay_seq

-- first ICU stay *for the current hospitalization*
, CASE
    WHEN DENSE_RANK() OVER (PARTITION BY ie.hadm_id ORDER BY ie.intime) = 1 THEN True
    ELSE False END AS first_icu_stay

FROM icustays ie
INNER JOIN admissions adm
    ON ie.hadm_id = adm.hadm_id
INNER JOIN patients pat
    ON ie.subject_id = pat.subject_id
WHERE adm.has_chartevents_data = 1
ORDER BY ie.subject_id, adm.admittime, ie.intime;
')

icustay_detail <- as_tibble(icustay_detail)

cohort <- icustay_detail %>%
  filter(admission_age >18 & admission_age < 90 & icustay_seq ==1 & admission_type == 'EMERGENCY')
dim(cohort)
names(cohort)

LabTests<-tbl(con,"d_labitems") %>%
  as_tibble()

# D_labitems table and corresponding to labevents table.
po_id  <- c(50971)
so_id <- c(50983)
bi_id <- c(50882)
wbc_id <- c(51301)
gl_id <- c(50931)
ch_id <- c(50902) #Chloride
he_id <- c(51221) #hematocrit
ma_id <- c(50960) #magnesium
ca_id <- c(50893) #calcium
ph_id <- c(50970) #phosphorus
la_id <- c(50813) #lactate

labevents_iv <- tbl(con,"labevents") %>%
```

```r
#  filter(subject_id %in% procedureevents_iv$subject_id) %>%
  filter(itemid == 50902 | itemid == 51221 | itemid == 50960 | itemid == 50893 |
           itemid == 50970 | itemid == 50813 | itemid == 50931 | itemid == 51301 |
           itemid == 50882 | itemid == 50983 | itemid == 50971)
LabE <- as_tibble(labevents_iv)

labs <- LabE %>%
  filter(subject_id %in% cohort$subject_id)

labs <- LabE %>%
  filter(subject_id %in% cohort$subject_id)
labs <- LabE %>%
  filter(subject_id %in% cohort$subject_id)
labs <- labs %>%
  group_by(subject_id,itemid) %>%
  arrange(charttime) %>%
  filter(row_number() == 1)
labs_w <- spread (labs,itemid,valuenum) %>%
  group_by(subject_id) %>%
  arrange(charttime) %>%
  filter(row_number() == 1)
#labs_w <- labs_w[complete.cases(labs_w),]
data <- cohort %>%
  left_join(labs_w,by="subject_id")

chartevents_iv <- tbl(con,"chartevents") %>%
  filter(itemid == 211 | itemid == 6)

chartevents <- as_tibble(chartevents_iv) %>%
  filter(icustay_id %in% cohort$icustay_id)
chartevents <- chartevents %>%
  group_by(icustay_id,itemid) %>%
  arrange(charttime) %>%
  filter(row_number() == 1)
chartevents_w <- spread (labs,itemid,valuenum) %>%
  group_by(icustay_id) %>%
  arrange(charttime) %>%
  filter(row_number() == 1)
#labs_w <- labs_w[complete.cases(labs_w),]
data <- cohort %>%
  left_join(chartevents_w,by="icustay_id")

dbDisconnect(con)
```

Data Analysis Code

```r
install.packages("pacman")
install.packages("gtsummary")
remotes::install_github("rstudio/gt")

require(pacman)
p_load(DBI, tidyverse, RPostgreSQL, h2o, missForest, kableExtra, xtable, gtsummary)
```

```r
Cohort <- readRDS("FinalDataUpdated.RDS")
Cohort <- as.data.frame(Cohort)[, -1]
Cohort  <- Cohort %>% mutate(
  gender = as.factor(gender),
  admission_type = as.factor(admission_type),
  first_careunit = as.factor(first_careunit),
  ethnicity = as.factor(ethnicity),
  dod = as.Date(as.character(dod)),
  dischtime = as.Date(as.character(dischtime)),
  THIRTY_DAY_MORT = ifelse(hospital_expire_flag == 'Y' | (dod - dischtime) < 30, 1, 0),
  hospital_expire_flag = as.factor(hospital_expire_flag),
  first_hosp_stay = as.factor(first_hosp_stay),
  THIRTY_DAY_MORT = as.factor(THIRTY_DAY_MORT)) %>%
  filter(!is.na(THIRTY_DAY_MORT)) %>%
  select(-hadm_id.x, -hadm_id.y, icustay_id, -dod, -admittime,
         -dischtime, -admission_type, -intime, -outtime, -icustay_seq,
         -first_icu_stay, -row_id, -charttime, -value, -valueuom, -flag,
         -icustay_id)

pMiss <- function(x){sum(is.na(x))/length(x)*100}

Var_Missing <- apply(Cohort, 2, pMiss)
Var_Dat <- data.frame(Name = names(Cohort),
                      Missing_Percentage = as.numeric(Var_Missing))
Patent_Missing <- apply(Cohort, 1, pMiss)
Patent_Dat <- data.frame(Name = 1:nrow(Cohort),
                         Missing_Percentage = as.numeric(Patent_Missing))

ggplot(Patent_Dat, aes(x = Missing_Percentage)) +
  geom_histogram() +
  labs(x = "Missing Percentage",
       title = "Missing Percentage of Patient") +
  theme(plot.title = element_text(hjust = 0.5))

ggplot(Var_Dat, aes(x = Name, y = Missing_Percentage)) +
  geom_bar(stat="identity") +
  labs(x = "Variables",
       y = "Missing Percentage",
       title = "Missing Percentage of Variables") +
  theme(axis.text.x = element_text(angle=90, vjust = 0.5),
        plot.title = element_text(hjust = 0.5))

Cohort[, -c(5, 10, 12:20)] %>%
  # build base summary table
  tbl_summary(
    by = THIRTY_DAY_MORT,
    # change statistics printed in table
    statistic = list(all_continuous() ~ "{mean} ({sd})",
                     all_categorical() ~ "{n} / {N} ({p}%)")
  ) %>%
  add_p(test = list(pvalue_fun = function(x) style_pvalue(x, digits = 2))) %>%
  # add statistic labels
  add_stat_label() %>%
  # bold variable labels, italicize levels
  bold_labels() %>%
  italicize_levels()

Cohort <- Cohort[, -c(5, 10, 12:20)]
Comp_Cohort <- Cohort[complete.cases(Cohort), ]
```

```
h2o.init(nthreads = -1,
         max_mem_size = "2G")
h2o.removeAll()

Temp <- as.h2o(Comp_Cohort)
splits <- h2o.splitFrame(
  Temp,           ##  splitting the H2O frame we read above
  c(0.6,0.2),    ##  create splits of 60% and 20%;
                 ##  H2O will create one more split of 1-(sum of these parameters)
                 ##  so we will get 0.6 / 0.2 / 1 - (0.6+0.2) = 0.6/0.2/0.2
  seed=1234)     ##  setting a seed will ensure reproducible results (not R's seed)

train <- h2o.assign(splits[[1]], "train.hex")
                 ## assign the first result the R variable train
                 ## and the H2O name train.hex
valid <- h2o.assign(splits[[2]], "valid.hex")   ## R valid, H2O valid.hex
test <- h2o.assign(splits[[3]], "test.hex")     ## R test, H2O test.hex
nfolds <- 5

rf1 <- h2o.randomForest(
  training_frame = train,                     ## Id of the training data frame
  validation_frame = valid,                   ## Id of the validation data frame
  x=1:9,                                      ## A vector containing the names or indices of the predictor
                                              ## variables to use in building the model
  y=10,                                       ## The name or column index of the response variable in the
  nfolds = nfolds,                            ## Number of folds for K-fold cross-validation. Defaults to
  fold_assignment = "Modulo",                 ## Cross-validation fold assignment scheme.
  keep_cross_validation_predictions = TRUE,## Logical, Whether to keep the predictions of the cross-val
  model_id = "rf_covType2",                   ##
  ntrees = 500,                               ## Number of trees., Defaults to 50
  max_depth = 30,                             ## Maximum tree depth. Defaults to 20
  stopping_rounds = 2,                        ## Early stopping based on convergence of stopping metric. S
                                              ## moving average of length k of the stpping metric does not
                                              ## k:=stopping rounds scoring events.
  stopping_tolerance = 1e-2,                  ## Relative tolerance for metric-based stopping criterion (s
                                              ## improvement is not at least this much)
  score_each_iteration = T,                   ## Logical. Whether to score during each iteration of model
  seed=3000000                                ## Seed for random numbers
)

rf_perf <- h2o.performance(rf1)

gbm1 <- h2o.gbm(
  training_frame = train,
  validation_frame = valid,
```

```r
  x=1:9,
  y=10,
  nfolds = nfolds,
  fold_assignment = "Modulo",
  keep_cross_validation_predictions = TRUE,
  ntrees = 500,
  learn_rate = 0.3,           ## Learning rate (from 0.0 to 1.0)
  max_depth = 10,
  sample_rate = 0.7,          ## use a random 70% of the rows to fit each tree
  col_sample_rate = 0.7,      ## use 70% of the columns to fit each tree
  stopping_rounds = 2,
  stopping_tolerance = 0.01,
  score_each_iteration = T,
  model_id = "gbm_covType3",
  seed = 2000000
)

gbm_perf <- h2o.performance(gbm1)

## Deep Learning
y <- "THIRTY_DAY_MORT"
x <- setdiff(names(train), y)
dnn1 <- h2o.deeplearning(
  x = x,
  y = y,
  training_frame = train,
  validation_frame = valid,
  distribution = "bernoulli",         ## Distribution function.
  epochs=10,                          ## How many times the dateset should be iterated, can be fraction
  stopping_metric="misclassification",## Metric to use for early stopping.
  stopping_tolerance=1e-2,
  stopping_rounds=2,
  score_validation_samples=10000,     ## Method used to sample validation dataset for scoring.
  score_duty_cycle=0.025,             ## Maximum duty cycle fraction for scoring.
  adaptive_rate=F,                    ## Logical. Adaptive learning rate.
  momentum_start=0.5,                 ## Initial momentum at the beginning of training.
  momentum_stable=0.9,                ## Final momentum after the ramp is over.
  momentum_ramp=1e7,                  ## Number of training samples for which momentum increases.
  l1=1e-5,                            ## L1 regularization
  l2=1e-5,                            ## L2 regularization
  activation=c("Rectifier"),          ## Activation function.
  max_w2=10,                          ## Constraint for squared sum of incoming weights per unit
  nfolds = nfolds,
  fold_assignment = "Modulo",
  keep_cross_validation_predictions = TRUE,
  seed = 1
)

dnn_perf <- h2o.performance(dnn1)

xgb1 <- h2o.xgboost(
  x = x,
  y = y,
  training_frame = train,
  validation_frame = valid,
  ntrees = 500,
  distribution = "bernoulli",
  max_depth = 8,
  min_rows = 1,
  learn_rate = 0.1,
  sample_rate = 0.7,
  col_sample_rate = 0.9,
  nfolds = nfolds,
  fold_assignment = "Modulo",
  keep_cross_validation_predictions = TRUE,
  seed = 1
)

xgb_perf <- h2o.performance(xgb1)

h2o.confusionMatrix(dnn_perf) %>%
  as.data.frame() %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Validation Results: Deep Neural Network (DNN)") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

h2o.confusionMatrix(gbm_perf) %>%
  as.data.frame() %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Validation Results: Gradient Boosting Model (GBM)") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

h2o.confusionMatrix(rf_perf) %>%
  as.data.frame() %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Validation Results: Random Forest (RF)") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

h2o.confusionMatrix(xgb_perf) %>%
  as.data.frame() %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Validation Results: xgboost") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

rf_auc <- h2o.auc(rf_perf)
gbm_auc <- h2o.auc(gbm_perf)
dnn_auc <- h2o.auc(dnn_perf)
xgb_auc <- h2o.auc(xgb_perf)
```

```r
data.frame(AUC = c(rf_auc, gbm_auc, dnn_auc, xgb_auc),
           Methods = c("RF", "GBM", "DNN", "xgboost")) %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Validation Results: AUC") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

rf_test <- h2o.performance(rf1, newdata = test)
gbm_test <- h2o.performance(gbm1, newdata = test)
dnn_test <- h2o.performance(dnn1, newdata = test)
xgb_test <- h2o.performance(xgb1, newdata = test)

h2o.confusionMatrix(dnn_test) %>%
  as.data.frame() %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Testing Results: Deep Neural Network (DNN)") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

h2o.confusionMatrix(gbm_test) %>%
  as.data.frame() %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Testing Results: Gradient Boosting Model (GBM)") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

h2o.confusionMatrix(rf_test) %>%
  as.data.frame() %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Testing Results: Random Forest (RF)") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

h2o.confusionMatrix(xgb_test) %>%
  as.data.frame() %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Testing Results: xgboost") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")


rf_test_auc <- h2o.auc(rf_test)
gbm_test_auc <- h2o.auc(gbm_test)
dnn_test_auc <- h2o.auc(dnn_test)
xgb_test_auc <- h2o.auc(xgb_test)

data.frame(AUC = c(rf_test_auc, gbm_test_auc, dnn_test_auc, xgb_test_auc),
           Methods = c("RF", "GBM", "DNN", "xgboost")) %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Testing Results: AUC") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

Imputed_Cohort <- missForest(Cohort)
Cohort_Imp <- Imputed_Cohort$ximp

Temp_Imp <- as.h2o(Cohort_Imp)
splits <- h2o.splitFrame(
  Temp_Imp,           ##  splitting the H2O frame we read above
  c(0.6,0.2),    ##  create splits of 60% and 20%;
                 ##  H2O will create one more split of 1-(sum of these parameters)
                 ##  so we will get 0.6 / 0.2 / 1 - (0.6+0.2) = 0.6/0.2/0.2
  seed=1234)     ##  setting a seed will ensure reproducible results (not R's seed)

train <- h2o.assign(splits[[1]], "train.hex")
                ## assign the first result the R variable train
                ## and the H2O name train.hex
valid <- h2o.assign(splits[[2]], "valid.hex")   ## R valid, H2O valid.hex
test <- h2o.assign(splits[[3]], "test.hex")     ## R test, H2O test.hex
nfolds <- 5

rf2 <- h2o.randomForest(        ##
  training_frame = train,       ##
  validation_frame = valid,
  x=1:9,                        ##
  y=10,
  nfolds = nfolds,
  fold_assignment = "Modulo",
  keep_cross_validation_predictions = TRUE,
  model_id = "rf_covType2",     ##
  ntrees = 500,                 ##
  max_depth = 30,               ## Increase depth, from 20
  stopping_rounds = 2,          ##
  stopping_tolerance = 1e-2,    ##
  score_each_iteration = T,     ##
  seed=3000000)

rf_perf <- h2o.performance(rf2)

gbm2 <- h2o.gbm(
  training_frame = train,     ##
  validation_frame = valid,   ##
  x=1:9,                     ##
  y=10,
  nfolds = nfolds,
  fold_assignment = "Modulo",
  keep_cross_validation_predictions = TRUE,##
  ntrees = 500,                ## add a few trees (from 20, though default is 50)
  learn_rate = 0.3,           ## increase the learning rate even further
  max_depth = 10,             ##
  sample_rate = 0.7,          ## use a random 70% of the rows to fit each tree
  col_sample_rate = 0.7,       ## use 70% of the columns to fit each tree
  stopping_rounds = 2,        ##
  stopping_tolerance = 0.01,  ##
  score_each_iteration = T,   ##
  model_id = "gbm_covType3",  ##
  seed = 2000000)             ##

gbm_perf <- h2o.performance(gbm2)

## Deep Learning
y <- "THIRTY_DAY_MORT"
x <- setdiff(names(train), y)
dnn2 <- h2o.deeplearning(
  x = x,
  y = y,
  training_frame = train,
  validation_frame = valid,
  distribution = "bernoulli",
  epochs=10,
  stopping_metric="misclassification",
  stopping_tolerance=1e-2,
  stopping_rounds=2,
  score_validation_samples=10000,
  score_duty_cycle=0.025,
  adaptive_rate=F,
  momentum_start=0.5,
  momentum_stable=0.9,
  momentum_ramp=1e7,
  l1=1e-5,
  l2=1e-5,
  activation=c("Rectifier"),
  max_w2=10,
  nfolds = nfolds,
  fold_assignment = "Modulo",
  keep_cross_validation_predictions = TRUE,
  seed = 1
)

dnn_perf <- h2o.performance(dnn2)

xgb2 <- h2o.xgboost(
  x = x,
  y = y,
  training_frame = train,
  validation_frame = valid,
  ntrees = 500,
  distribution = "bernoulli",
  max_depth = 8,
  min_rows = 1,
  learn_rate = 0.1,
  sample_rate = 0.7,
  col_sample_rate = 0.9,
  nfolds = nfolds,
  fold_assignment = "Modulo",
  keep_cross_validation_predictions = TRUE,
  seed = 1
)

xgb_perf <- h2o.performance(xgb2)

h2o.confusionMatrix(dnn_perf) %>%
  as.data.frame() %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Validation Results (Imputation): Deep Neural Network (DNN)") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

h2o.confusionMatrix(gbm_perf) %>%
  as.data.frame() %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Validation Results (Imputation): Gradient Boosting Model (GBM)") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

h2o.confusionMatrix(rf_perf) %>%
  as.data.frame() %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Validation Results (Imputation): Random Forest (RF)") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

h2o.confusionMatrix(xgb_perf) %>%
  as.data.frame() %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Validation Results (Imputation): xgboost") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

rf_auc <- h2o.auc(rf_perf)
gbm_auc <- h2o.auc(gbm_perf)
dnn_auc <- h2o.auc(dnn_perf)
xgb_auc <- h2o.auc(xgb_perf)

data.frame(AUC = c(rf_auc, gbm_auc, dnn_auc, xgb_auc),
           Methods = c("RF", "GBM", "DNN", "xgboost")) %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Validation Results (Imputation): AUC") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")
```

```r
rf_test <- h2o.performance(rf2, newdata = test)
gbm_test <- h2o.performance(gbm2, newdata = test)
dnn_test <- h2o.performance(dnn2, newdata = test)
xgb_test <- h2o.performance(xgb2, newdata = test)

h2o.confusionMatrix(dnn_test) %>%
  as.data.frame() %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Testing Results (Imputation): Deep Neural Network (DNN)") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

h2o.confusionMatrix(gbm_test) %>%
  as.data.frame() %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Testing Results (Imputation): Gradient Boosting Model (GBM)") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

h2o.confusionMatrix(rf_test) %>%
  as.data.frame() %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Testing Results (Imputation): Random Forest (RF)") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

h2o.confusionMatrix(xgb_test) %>%
  as.data.frame() %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Testing Results (Imputation): xgboost") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")


rf_test_auc <- h2o.auc(rf_test)
gbm_test_auc <- h2o.auc(gbm_test)
dnn_test_auc <- h2o.auc(dnn_test)
xgb_test_auc <- h2o.auc(xgb_test)

data.frame(AUC = c(rf_test_auc, gbm_test_auc, dnn_test_auc, xgb_test_auc),
           Methods = c("RF", "GBM", "DNN", "xgboost")) %>%
  kable(format = "html",
        booktabs = TRUE,
        caption = "Testing Results (Imputation): AUC") %>%
  kable_styling(position = "center",
                latex_options = "hold_position")

h2o.shutdown(prompt = FALSE)
```

References

Feng, Mengling, Jakob I McSparron, Dang Trung Kien, David J Stone, David H Roberts, Richard M Schwartzstein, Antoine Vieillard-Baron, and Leo Anthony Celi. 2018. "Transthoracic Echocardiography and Mortality in Sepsis: Analysis of the Mimic-Iii Database." *Intensive Care Medicine* 44 (6). Springer: 884–92.

Harutyunyan, Hrayr, Hrant Khachatrian, David C Kale, Greg Ver Steeg, and Aram Galstyan. 2017. "Multitask Learning and Benchmarking with Clinical Time Series Data." *arXiv Preprint arXiv:1703.07771*.

Johnson, Alistair EW, and Roger G Mark. 2017. "Real-Time Mortality Prediction in the Intensive Care Unit." In *AMIA Annual Symposium Proceedings*, 2017:994. American Medical Informatics Association.

Sadeghi, Reza, Tanvi Banerjee, and William Romine. 2018. "Early Hospital Mortality Prediction Using Vital Signals." *Smart Health* 9. Elsevier: 265–74.

Silva, Ikaro, George Moody, Daniel J Scott, Leo A Celi, and Roger G Mark. 2012. "Predicting in-Hospital Mortality of Icu Patients: The Physionet/Computing in Cardiology Challenge 2012." In *2012 Computing in Cardiology*, 245–48. IEEE.

Vincent, Jean-Louis, Nathan D Nielsen, Nathan I Shapiro, Margaret E Gerbasi, Aaron Grossman, Robin Doroff, Feng Zeng, Paul J Young, and James A Russell. 2018. "Mean Arterial Pressure and Mortality in Patients with Distributive Shock: A Retrospective Analysis of the Mimic-Iii Database." *Annals of Intensive Care* 8 (1). Springer: 107.