Gradient boost tree model with catboost

Code ▾

load requied package

Hide

```
pacman::p_load(tidyr, dplyr, ggplot2, catboost, caret, plotROC, tensorflow)
```

set working directory

load required data

Hide

```
train_over <- read.csv("train_over.csv")
train_under <- read.csv("train_under.csv")
train_both <- read.csv("train_both.csv")
train_rose <- read.csv("train_rose.csv")
test <- read.csv("test.csv")
train0 <- read.csv("train0.csv")
```

custom function

Hide

```
# compute model accuracy
calc_accuracy <- function(prediction, expected, threshold) {
  labels <- ifelse(prediction > threshold, 1, 0)
  accuracy <- sum(labels == expected) / length(labels)
  return(accuracy)
}
```

remove first column

Hide

```
train_over <- train_over[,c(-1)]
train_under <- train_under[,c(-1)]
train_both <- train_both[,c(-1)]
train_rose <- train_rose[,c(-1)]
test <- test[,c(-1)]
train0 <- train0[,c(-1)]
```

generate cat pool

Hide

```
column_description_vector <-  colnames(train_over)
target <- 12

train_over_pool <- catboost.load_pool(data=train_over[,-target], label = train_over[,target])
test_pool <- catboost.load_pool(data=test[,-target], label = test[,target])
```

train model - initial iter

Hide

```
path <- c("C:/Users/Shenc/Documents/NUS EBAC/EBA5005/CA/Model")

fit_params <- list(task_type="GPU",
                   loss_function = "Logloss",
                   iterations = 150,
                   learning_rate = 0.3,
                   random_seed = 101,
                   l2_leaf_reg = 5,
                   bagging_temperature = 3,
                   #sampling_frequency = "PerTree",
                   #ignored_features = c(4,9),
                   border_count = 32,
                   depth = 3,
                   leaf_estimation_method = "Newton",
                   feature_border_type = "GreedyLogSum",
                   thread_count = 500,
                   logging_level = 'Silent',
                   train_dir = path,
                   od_type = "Iter")

model_over <- catboost.train(train_over_pool, test_pool, fit_params)

#tensorboard(log_dir = path)
```

grid search

Hide

```
# drop_columns <- "hospital_death"
# x <- train_under[,!(names(train_over) %in% drop_columns)]
# y <- train_under[,c("hospital_death")]
#
# fit_control <- trainControl(method = "cv",
#                             number = 5,
#                             classProbs = TRUE)
#
# #seq(0.01,0.1, by=0.01)
# #seq(100,1000, by = 50)
#
# # set grid options
# grid <- expand.grid(
#   depth = (3:7),
#   learning_rate = 0.04,
#   iterations = 150,
#   l2_leaf_reg = 4,
#   rsm = 0.95,
#   border_count = 32
# )
#
# model <- caret::train(x, as.factor(make.names(y)),
#                 method = catboost.caret,
#                 logging_level = 'Silent', preProc = NULL,
#                 tuneGrid = grid, trControl = fit_control)
#
# print(model)
#
# importance <- varImp(model, scale = FALSE)
# print(importance)
```

Predict and evaluate

Hide

```
prediction <- catboost.predict(model_over, test_pool, prediction_type = 'Probability')
# cat("Sample predictions: ", sample(prediction, 5), "\n")
```

confusion matrix

Hide

```
# test set confusion matrix
test_matrix <- catboost.predict(model_over, test_pool, prediction_type = 'Class')
table(test[,target], test_matrix)
```

```
   test_matrix
        0     1
  0 17358  4649
  1   370  1664
```

Hide

```
# train set confusion matrix
train_matrix <- catboost.predict(model_over, train_over_pool, prediction_type = 'Class')
table(train_over[,target], train_matrix)
```

```
   train_matrix
        0     1
  0 40623 10727
  1  9069 51861
```

Hide

```
# works properly only for Logloss
accuracy <- calc_accuracy(prediction, test[,target], 0.493472)
cat("\nAccuracy: ", accuracy, "\n")
```

```

Accuracy:  0.7886527
```

ROC and AUC

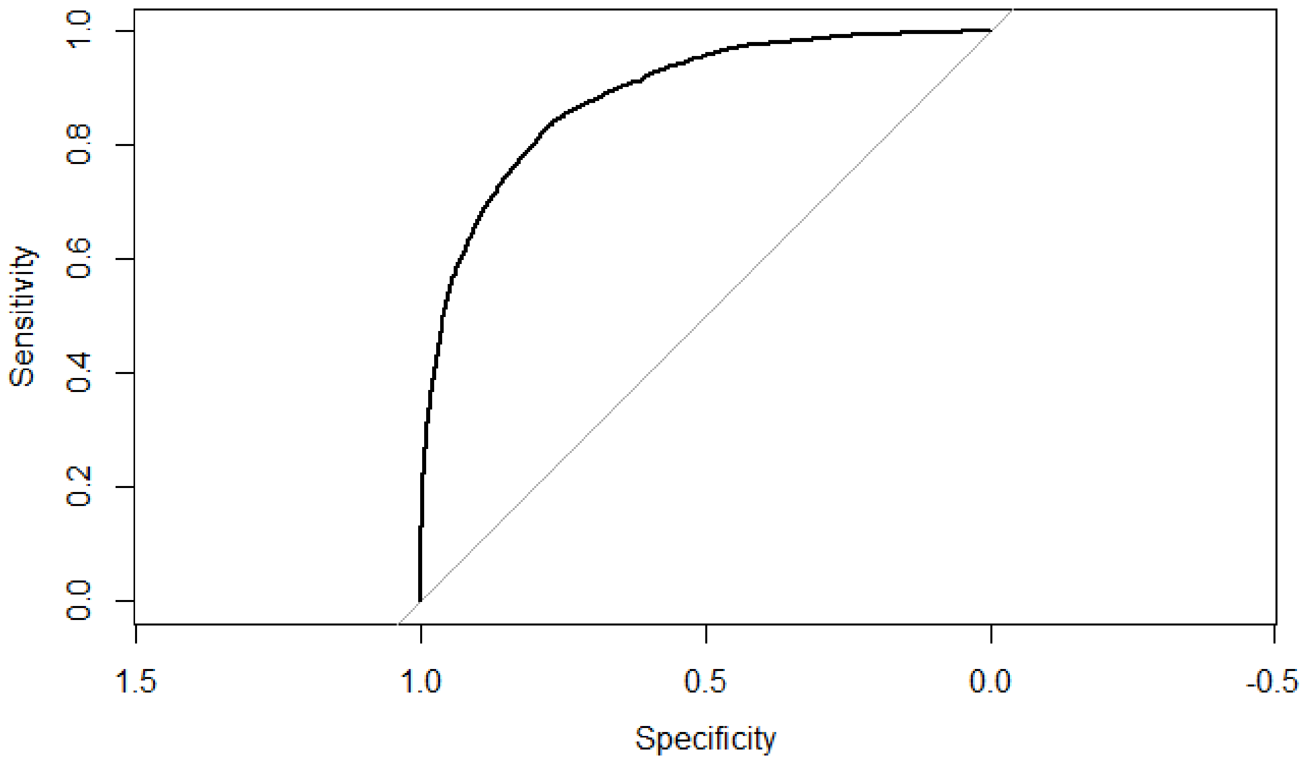
Hide

```
roc_obj <- pROC::roc(test$hospital_death, prediction)
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```

Hide

```
plot(roc_obj)
```



Hide

```
pROC::auc(roc_obj)
```

```
Area under the curve: 0.8867
```

Hide

```
pROC::coords(roc_obj, "best", "threshold")
```

threshold	specificity	sensitivity
<dbl>	<dbl>	<dbl>
0.493472	0.7852501	0.8254671

1 row

feature importance

Hide

```
# cat("\nFeature importances", "\n")
feature_imp <- catboost.get_feature_importance(model_over, train_over_pool)
feature_df <- data.frame(columnNameILike = row.names(feature_imp), feature_imp)
colnames(feature_df) <- c("feature", "importance")

# find features with 0 importance
least_imp <- feature_df %>%
  filter(importance == 0) %>%
  select(feature)
```

remove features with importance = 0

prepare cat pool for under sampling dataset

Hide

```
# compute accuracy
accuracy_under <- calc_accuracy(prediction_under, test[,target], 0.462345)
cat("\nAccuracy: ", accuracy_under, "\n")
```

```

Accuracy:  0.7928123
```

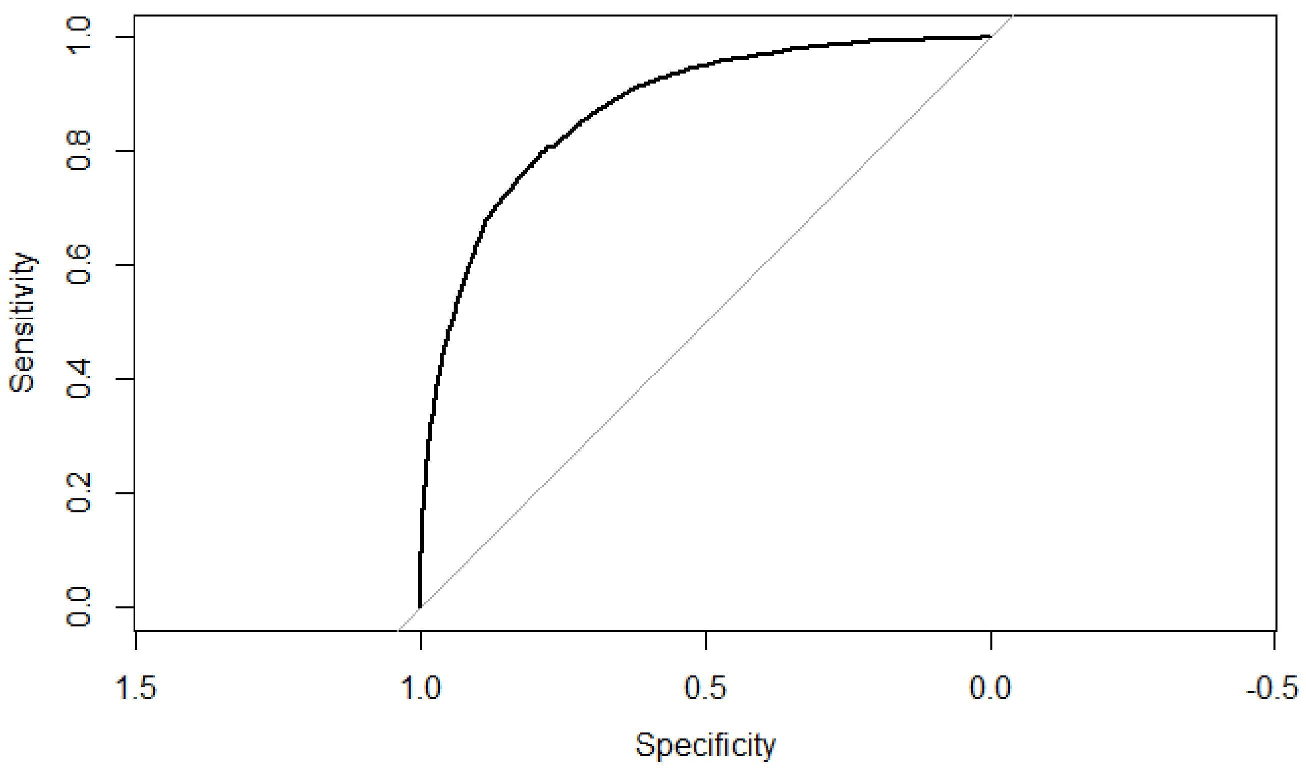
Hide

```
# feature importance
feature_imp1 <- catboost.get_feature_importance(model_under, train_under_pool)
feature_df1 <- data.frame(columnNameILike = row.names(feature_imp1), feature_imp1)
colnames(feature_df1) <- c("feature", "importance")

# identify and remove 0 importance features
least_imp1 <- feature_df1 %>%
  filter(importance == 0) %>%
  select(feature)

# model threshold
roc_obj1 <- pROC::roc(test$hospital_death, prediction_under)
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```

Hide

```
plot(roc_obj1)
```



Hide

```
pROC::auc(roc_obj1)
```

```
Area under the curve: 0.8748
```

Hide

```
pROC::coords(roc_obj1, "best", "threshold")
```

```
The 'transpose' argument to FALSE by default since pROC 1.16. Set transpose = TRUE explicitly
to revert to the previous behavior, or transpose = TRUE to silence this warning. Type help(co
ords_transpose) for additional information.
```

threshold	specificity	sensitivity
<dbl>	<dbl>	<dbl>
0.4623464	0.7928841	0.7920354

1 row

prepare cat pool for over and under sampling dataset

Hide

```
cat("\nAccuracy: ", accuracy_both, "\n")
```

```


Accuracy:  0.8091178
```

prepare cat pool for ROSE sampling dataset

Hide

```
# parameters tuning
fit_params3 <- list(task_type="GPU",
                   loss_function = "Logloss",
                   iterations = 150,
                   learning_rate = 0.04,
                   random_seed = 101,
                   l2_leaf_reg = 3,
                   bagging_temperature = 6,
                   #sampling_frequency = "PerTree",
                   #ignored_features = c(4,9),
                   border_count = 32,
                   depth = 3,
                   leaf_estimation_method = "Newton",
                   feature_border_type = "MinEntropy",
                   thread_count = 500,
                   logging_level = 'Silent',
                   train_dir = path,
                   od_type = "Iter")

# split train and test sets
train_rose_pool <- catboost.load_pool(data=train_rose[,-target], label = train_rose[,target])
model_rose<- catboost.train(train_rose_pool, test_pool, fit_params3)
prediction_rose <- catboost.predict(model_rose, test_pool, prediction_type = 'Probability')

# test set confusion matrix
test_matrix_rose <- catboost.predict(model_rose, test_pool, prediction_type = 'Class')
table(test[,target], test_matrix_rose)
```

```
   test_matrix_rose
        0     1
  0 19236  2771
  1   671  1363
```

Hide

```
# train set confusion matrix
train_matrix_rose <- catboost.predict(model_rose, train_rose_pool, prediction_type = 'Class')
table(train_rose[,target],train_matrix_rose)
```

```
   train_matrix_rose
        0     1
  0 23680  4329
  1  4712 23375
```

Hide

```
# compute accuracy
accuracy_rose <- calc_accuracy(prediction_rose, test[,target], 0.448931)
cat("\nAccuracy: ", accuracy_both, "\n")
```

```


Accuracy:  0.8091178
```

Hide

```
# feature importance
feature_imp3 <- catboost.get_feature_importance(model_rose, train_rose_pool)
feature_df3 <- data.frame(columnNameILike = row.names(feature_imp3), feature_imp3)
colnames(feature_df3) <- c("feature", "importance")

# identify and remove 0 importance features
least_imp3 <- feature_df3 %>%
  filter(importance == 0) %>%
  select(feature)

roc_obj <- pROC::roc(test$hospital_death, prediction_rose)
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```

Hide

```
pROC::auc(roc_obj)
```

```
Area under the curve: 0.8705
```

Hide

```
pROC::coords(roc_obj, "best", "threshold")
```

```
The 'transpose' argument to FALSE by default since pROC 1.16. Set transpose = TRUE explicitly
to revert to the previous behavior, or transpose = TRUE to silence this warning. Type help(co
ords_transpose) for additional information.
```

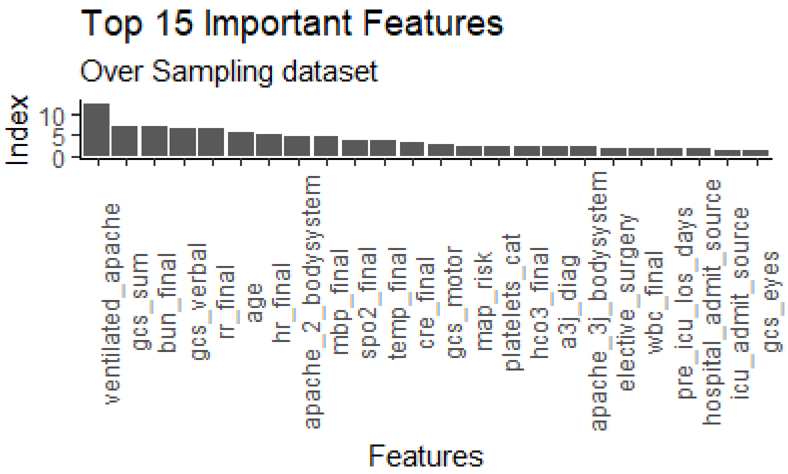
threshold <dbl>	specificity <dbl>	sensitivity <dbl>
0.4489328	0.8411415	0.7330383

1 row

feature importance plot

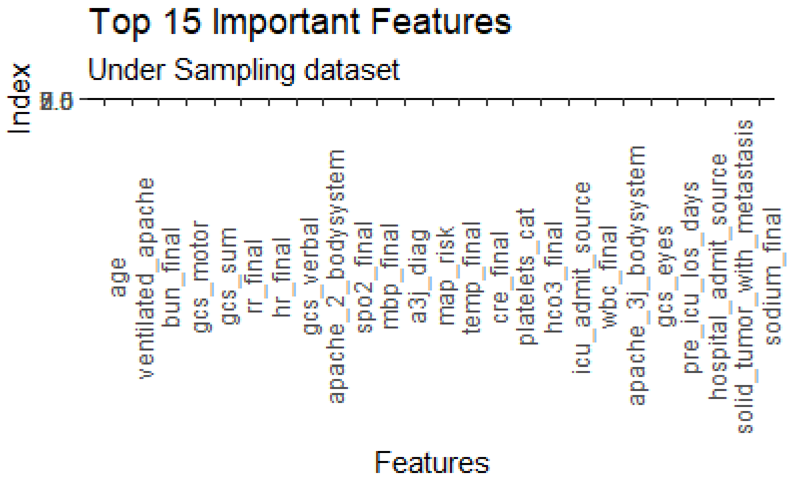
Hide

```
feature_df %>%
  arrange(importance) %>%
  top_n(importance, 15) %>%
  ggplot(aes(x = reorder(feature, -importance), y = importance)) +
  geom_col() +
  labs(title = "Top 15 Important Features",
       subtitle = "Over Sampling dataset",
       x = "Features",
       y = "Index") +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 90))
```



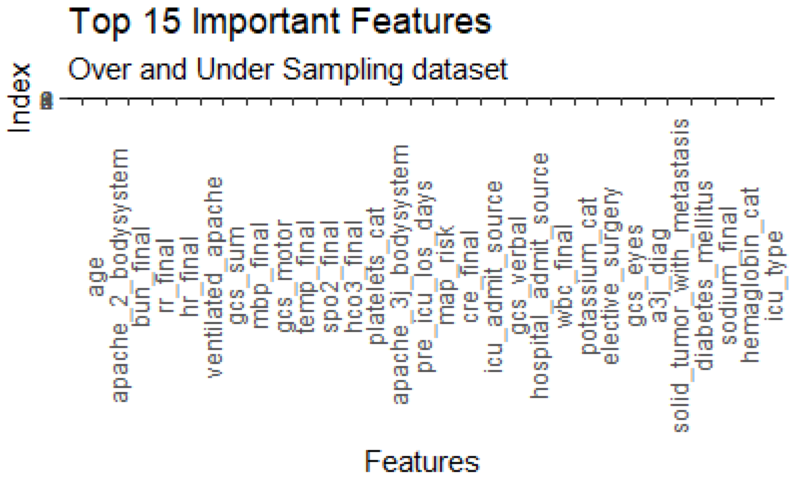
Hide

```
feature_df1 %>%
  arrange(importance) %>%
  top_n(importance, 15) %>%
  ggplot(aes(x = reorder(feature, -importance), y = importance)) +
  geom_col() +
  labs(title = "Top 15 Important Features",
       subtitle = "Under Sampling dataset",
       x = "Features",
       y = "Index") +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 90))
```



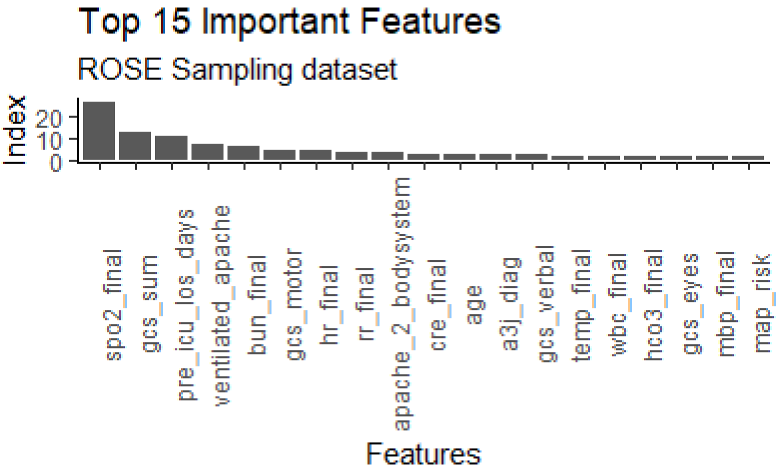
Hide

```
feature_df2 %>%
  arrange(importance) %>%
  top_n(importance, 15) %>%
  ggplot(aes(x = reorder(feature, -importance), y = importance)) +
  geom_col() +
  labs(title = "Top 15 Important Features",
       subtitle = "Over and Under Sampling dataset",
       x = "Features",
       y = "Index") +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 90))
```



Hide

```
feature_df3 %>%
  arrange(importance) %>%
  top_n(importance, 15) %>%
  ggplot(aes(x = reorder(feature, -importance), y = importance)) +
  geom_col() +
  labs(title = "Top 15 Important Features",
       subtitle = "ROSE Sampling dataset",
       x = "Features",
       y = "Index") +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 90))
```



features with 0 importance

Hide

```
least_imp
```

feature <fctr>
apache_post_operative
aids
leukemia
lymphoma
hto_final

5 rows

Hide

```
least_imp1
```

feature <fctr>
apache_post_operative

feature <fctr>
arf_apache
aids
hepatic_failure

4 rows

Hide

```
least_imp2
```

feature <fctr>
aids
leukemia

2 rows

Hide

```
least_imp3
```

feature <fctr>
apache_post_operative
arf_apache
gender
icu_stay_type
aids
cirrhosis
diabetes_mellitus
hepatic_failure
leukemia
lymphoma

1-10 of 12 rows Previous **1** 2 Next

for original train dataset (imbalanced)

Hide

```
# split train and test sets
train_imb_pool <- catboost.load_pool(data=train0[,-target], label = train0[,target])
model_imb<- catboost.train(train_imb_pool, test_pool, fit_params)
prediction_imb <- catboost.predict(model_imb, test_pool, prediction_type = 'Probability')

# test set confusion matrix
test_matrix_imb <- catboost.predict(model_imb, test_pool, prediction_type = 'Class')
table(test[,target], test_matrix_imb)
```

```
   test_matrix_imb
        0     1
  0 21750   257
  1  1435   599
```

Hide

```
# train set confusion matrix
train_matrix_imb <- catboost.predict(model_imb, train_imb_pool, prediction_type = 'Class')
table(train0[,target],train_matrix_imb)
```

```
   train_matrix_imb
        0     1
  0 50758   592
  1  3346  1400
```

Hide

```
# compute accuracy
accuracy_imb <- calc_accuracy(prediction_imb, test[,target])
cat("\nAccuracy: ", accuracy_both, "\n")
```

```

Accuracy:  0.8159394
```