```
library(data.table)
library(foreach)
library(readr)

blogData_train <- read_csv("blogData_train.csv")



View(blogData_train)
# retrieve filenames of test sets
test_filenames = list.files(pattern = "blogData_test")
# load and combine dataset
train = fread("blogData_train.csv")
test = foreach(i = 1:length(test_filenames), .combine = rbind) %do% {
temp = fread(test_filenames[i], header = F)
}
# log-transform
train[, V281 := log(1 + V281)]
test[, V281 := log(1 + V281)]

drop = c(8, 13, 28, 33, 38, 40, 43, 50, 278)
train[, (drop) := NULL]
test[, (drop) := NULL]
# write to files
write.csv(train, "BlogFeedback-Train.csv", row.names = F)
write.csv(test, "BlogFeedback-Test.csv", row.names = F)

### Basic Models ###
library(data.table)
library(MatrixModels)
library(e1071)
library(FNN)
library(glmnet)

library(ranger)
library(xgboost)
# load and combine dataset
train = fread("BlogFeedback-Train.csv")
test = fread("BlogFeedback-Test.csv")
# error measure
mse = function(y_hat, y) {
mse = mean((y - y_hat)^2)
return(mse)
}
# create design matrices
train_x = model.Matrix(V281 ~ . - 1, data = train, sparse = F)
train_x_sparse = model.Matrix(V281 ~ . - 1, data = train, sparse = T)
train_y = train$V281
test_x = model.Matrix(V281 ~ . - 1, data = test, sparse = F)
test_y = test$V281
train_xgb = xgb.DMatrix(data = as.matrix(train_x), label = train_y)
test_xgb = xgb.DMatrix(data = as.matrix(test_x), label = test_y)
# try kNN
pred_knn = knn.reg(train_x, test_x, train_y, k = 19)$pred
```

```
mse(pred_knn, test_y)
## [1] 0.6383154
# try LASSO
mdl_lasso = cv.glmnet(train_x_sparse, train_y, family = "gaussian", alpha
= 1)
pred_lasso = predict(mdl_lasso, newx = test_x)
mse(pred_lasso, test_y)
## [1] 0.6365795
# try random forest
mdl_rf = ranger(V281 ~ ., data = train, num.trees = 1000, mtry = 120,
write.forest = T)

pred_rf = predict(mdl_rf, test)
mse(pred_rf$predictions, test_y)
## [1] 0.397223
# try SVM
mdl_svm = svm(V281 ~ V52 + V55 + V61 + V51 + V54 + V21 + V6 + V10, data =
train, kernel = "radial", cost = 2, gamma = 0.25)
pred_svm = predict(mdl_svm, test)
mse(pred_svm, test_y)
## [1] 0.4454636
# try XGboost
mdl_xgb = xgboost(data = train_xgb, nround = 750, nthread = 4, max_depth =
6,eta = 0.025, subsample = 0.7, gamma = 3)


pred_xgb = predict(mdl_xgb, test_xgb)
mse(pred_xgb, test_y)
## [1] 0.3820247
### Stacked Generalization ###
library(data.table)
library(foreach)
library(MatrixModels)
library(e1071)
library(FNN)
library(glmnet)
library(ranger)

library(xgboost)
# load and combine dataset
train = fread("BlogFeedback-Train.csv")
test = fread("BlogFeedback-Test.csv")
# error measure
mse = function(y_hat, y) {
mse = mean((y - y_hat)^2)
return(mse)
}
# create design matrices
test_x = model.Matrix(V281 ~ . - 1, data = test, sparse = F)
test_x_sparse = model.Matrix(V281 ~ . - 1, data = test, sparse = T)
#test_xgb = xgb.DMatrix(data = as.matrix(test_x), label = test_y)
train_y = train$V281
test_y = test$V281
# divide training set into k folds
```

```
k = 5
cv_index = 1:nrow(train)
cv_index_split = split(cv_index, cut(seq_along(cv_index), k, labels =
FALSE))
# meta features from kNN
meta_knn_test = rep(0, nrow(test))
meta_knn_train = foreach(i = 1:k, .combine = c) %do% {
# split the raining set into two disjoint sets
train_index = setdiff(1:nrow(train), cv_index_split[[i]])
train_set1 = model.Matrix(V281 ~ . - 1, data = train[train_index], sparse
=
T)
train_set2 = model.Matrix(V281 ~ . - 1, data = train[cv_index_split[[i]]],
sparse = T)
# level 0 prediction
meta_pred = knn.reg(train_set1, train_set2, train[train_index]$V281, k =
19
)$pred
meta_knn_test = meta_knn_test + knn.reg(train_set1, test_x_sparse,
train[tr
ain_index]$V281, k = 19)$pred / k
return(meta_pred)
}
# meta features from LASSO
meta_glm_test = rep(0, nrow(test))
meta_glm_train = foreach(i = 1:k, .combine = c) %do% {
# split the raining set into two disjoint setstrain_index =
setdiff(1:nrow(train), cv_index_split[[i]])
train_set1 = model.Matrix(V281 ~ . - 1, data = train[train_index], sparse
=
T)
train_set2 = model.Matrix(V281 ~ . - 1, data = train[cv_index_split[[i]]],
sparse = T)
# level 0 prediction
temp_glm = cv.glmnet(train_set1, train[train_index]$V281, family =
"gaussia
n", alpha = 1)
meta_pred = predict(temp_glm, newx = train_set2)
meta_glm_test = meta_glm_test + predict(temp_glm, newx = test_x_sparse) /
k
return(meta_pred)
}
# meta features from SVM
meta_svm_test = rep(0, nrow(test))
meta_svm_train = foreach(i = 1:k, .combine = c) %do% {
# split the raining set into two disjoint sets
train_index = setdiff(1:nrow(train), cv_index_split[[i]])
train_set1 = train[train_index]
train_set2 = train[cv_index_split[[i]]]
# level 0 prediction
temp_svm = svm(V281 ~ V52 + V55 + V61 + V51 + V54 + V21 + V6 + V10, data =
train_set1,
kernel = "radial", cost = 2, gamma = 0.25)
meta_pred = predict(temp_svm, train_set2)
meta_svm_test = meta_svm_test + predict(temp_svm, test) / k
return(meta_pred)
}
# meta features from random forest
meta_rf_test = rep(0, nrow(test))
meta_rf_train = foreach(i = 1:k, .combine = c) %do% {
# split the raining set into two disjoint sets
train_index = setdiff(1:nrow(train), cv_index_split[[i]])
train_set1 = train[train_index]
train_set2 = train[cv_index_split[[i]]]
# level 0 prediction
temp_rf = ranger(V281 ~ ., data = train_set1, num.trees = 500, mtry = 120,
write.forest = T)
meta_pred = predict(temp_rf, train_set2)$predictions
meta_rf_test = meta_rf_test + predict(temp_rf, test)$predictions /
kreturn(meta_pred)
}


# meta features from XGBoost
meta_xgb_test = rep(0, nrow(test))
meta_xgb_train = foreach(i = 1:k, .combine = c) %do% {
# split the raining set into two disjoint sets
train_index = setdiff(1:nrow(train), cv_index_split[[i]])
train_set1 = model.Matrix(V281 ~ . - 1, data = train[train_index], sparse
=F)
train_set2 = model.Matrix(V281 ~ . - 1, data =
train[cv_index_split[[i]]],sparse = F)
# xgb data
train_set1_xgb = xgb.DMatrix(data = as.matrix(train_set1), label =
train[train_index]$V281)
train_set2_xgb = xgb.DMatrix(data = as.matrix(train_set2), label =
train[cv_index_split[[i]]]$V281)

temp_xgb = xgboost(data = train_set1_xgb, nround = 750, nthread = 4,
max_depth = 6, eta = 0.025, subsample = 0.7, gamma = 3)
meta_pred = predict(temp_xgb, train_set2_xgb)
meta_xgb_test = meta_xgb_test + predict(temp_xgb, test_xgb) / k
return(meta_pred)
}




# combine meta features
sg_col = c("meta_knn", "meta_glm", "meta_svm", "meta_rf", "meta_xgb", "y")
train_sg = data.frame(meta_knn_train, meta_glm_train, meta_svm_train,
meta_rf
_train, meta_xgb_train, train_y)
test_sg = data.frame(meta_knn_test, meta_glm_test, meta_svm_test,
meta_rf_tes
t, meta_xgb_test, test_y)
colnames(train_sg) = sg_col
colnames(test_sg) = sg_col
```

```
# ensemble with elastic-net regression
train_sg_sparse = model.Matrix(y ~ . - 1, data = train_sg, sparse = T)
test_sg_sparse = model.Matrix(y ~ . - 1, data = test_sg, sparse = T)
mdl_glm = cv.glmnet(train_sg_sparse, train_y, family = "gaussian", alpha =
0.
2)
pred_glm = predict(mdl_glm, newx = test_sg_sparse, s = "lambda.min")
mse(pred_glm, test_y)
## [1] 0.3840147
library(data.table)
library(foreach)
library(MatrixModels)
library(xgboost)
library(ranger)
# load and combine dataset
train = fread("BlogFeedback-Train.csv")
test = fread("BlogFeedback-Test.csv")
# error measure
mse = function(y_hat, y) {
mse = mean((y - y_hat)^2)return(mse)}
# create design matrices
train_x = model.Matrix(V281 ~ . - 1, data = train, sparse = F)
train_y = train$V281
test_x = model.Matrix(V281 ~ . - 1, data = test, sparse = F)
test_y = test$V281
train_xgb = xgb.DMatrix(data = as.matrix(train_x), label = train_y)
test_xgb = xgb.DMatrix(data = as.matrix(test_x), label = test_y)
# number of models
n = 5
# fit XGBoost
pred_xgb = foreach(i = 1:n, .combine = cbind) %do% { mdl_xgb =
xgboost(data= train_xgb, nround = 750, nthread = 4, max_depth = 6, eta =
0.025, subsample= 0.7, gamma = 3)
return(predict(mdl_xgb, test_xgb))}
# fit random forest
pred_rf = foreach(i = 1:n, .combine = cbind) %do% { mdl_rf = ranger(V281 ~
.,data = train, num.trees = 1000, mtry = 120, write.forest = T)
return(predict(mdl_rf, test)$predictions)}
# weighted average
mse(rowMeans(pred_rf) * 0.25 + rowMeans(pred_xgb) * 0.75, test_y)
```

Basic testing model

Model	Error
k-NN	0.638315
LASSO	0.635483
SVM	0.445464
Random Forest	0.396741
Boosting	0.382399