Bike sharing

Load default libraries

```
library(data.table)
library(dplyr)
library(lubridate)
library(ggplot2)
library(caret)
library(stringi)
library(xgboost)
library(reshape2)
theme_set(theme_bw())
set.seed(42)
```

Read the data

```
full_train = fread("./train.csv", header = T, sep = ",", integer64 = "numeric")
test = fread("./test.csv", header = T, sep = ",", integer64 = "numeric")
```

Change type to categorical variables

```
fix_types = function(data) {
  data %>%
    mutate(
      datetime = ymd_hms(datetime),
      workingday = 1 - workingday,
      weather = weather) %>%
    mutate(
      year = year(datetime),
      month = month(datetime),
      day = as.character(mday(datetime)),
      wday = wday(datetime),
      hour = hour(datetime),
      hour4 = as.character(round(hour(datetime) / 4)),
      hour3 = as.character(round(hour(datetime) / 3)),
      hour6 = as.character(round(hour(datetime) / 6)),
      hour8 = as.character(round((hour(datetime) - 4) / 8)),
      hour12 = as.character(round((hour(datetime)) / 12)),
      month2  = as.character(round((month(datetime)) / 2)))
}
full_train = fix_types(full_train)
test = fix_types(test)

train = full_train[as.integer(full_train$day) < 13]
validate = full_train[as.integer(full_train$day) >= 13]
```

Prepare features

Prepare target

We going to predict log(Y+1) to optimize the target cost function

```
preparedTrainTarget = log(train$count + 1)

preparedValidateTarget = log(validate$count + 1)
preparedFullTarget =log(full_train$count + 1)
```

Xgboost train and cross-validate

Interesting link: how to tune hyperparameters

```
dtrain <- xgb.DMatrix(train_matrix, label = preparedTrainTarget)

xgbControl = list(
   subsample=0.8, colsample_bytree = 0.8, metrics=list("rmse"), gamma = 0.9,
                  max.depth = 6, eta = 0.11, alpha = 1, lambda = 1, objective = "reg:linear"
)
model = xgboost(params = xgbControl, data = dtrain,
                nround=2000, nthread = 4, print.every.n = 500)
```

```
## [0]  train-rmse:3.848741
## [500]    train-rmse:0.233766
## [1000]   train-rmse:0.229112
## [1500]   train-rmse:0.226564
```

```
history <- xgb.cv(params = xgbControl, data = dtrain,
                  nround=2000, nthread = 4, nfold = 10, print.every.n = 500)
```

```
## [0]  train-rmse:3.849032+0.002681    test-rmse:3.849260+0.025239
## [500]    train-rmse:0.234618+0.001421    test-rmse:0.286852+0.015078
## [1000]   train-rmse:0.230440+0.001408    test-rmse:0.284658+0.014445
## [1500]   train-rmse:0.228490+0.001286    test-rmse:0.283699+0.014335
```

```
print(tail(history))
```

```
##    train.rmse.mean train.rmse.std test.rmse.mean test.rmse.std
## 1:        0.227079       0.001799       0.283079      0.014174
## 2:        0.227066       0.001811       0.283069      0.014165
## 3:        0.227066       0.001811       0.283069      0.014167
## 4:        0.227066       0.001811       0.283068      0.014166
## 5:        0.227066       0.001811       0.283066      0.014171
## 6:        0.227066       0.001811       0.283063      0.014168
```

Validate Score

```
validate_predictions = predict(model, validate_matrix)
RMSE(validate_predictions, preparedValidateTarget)
```

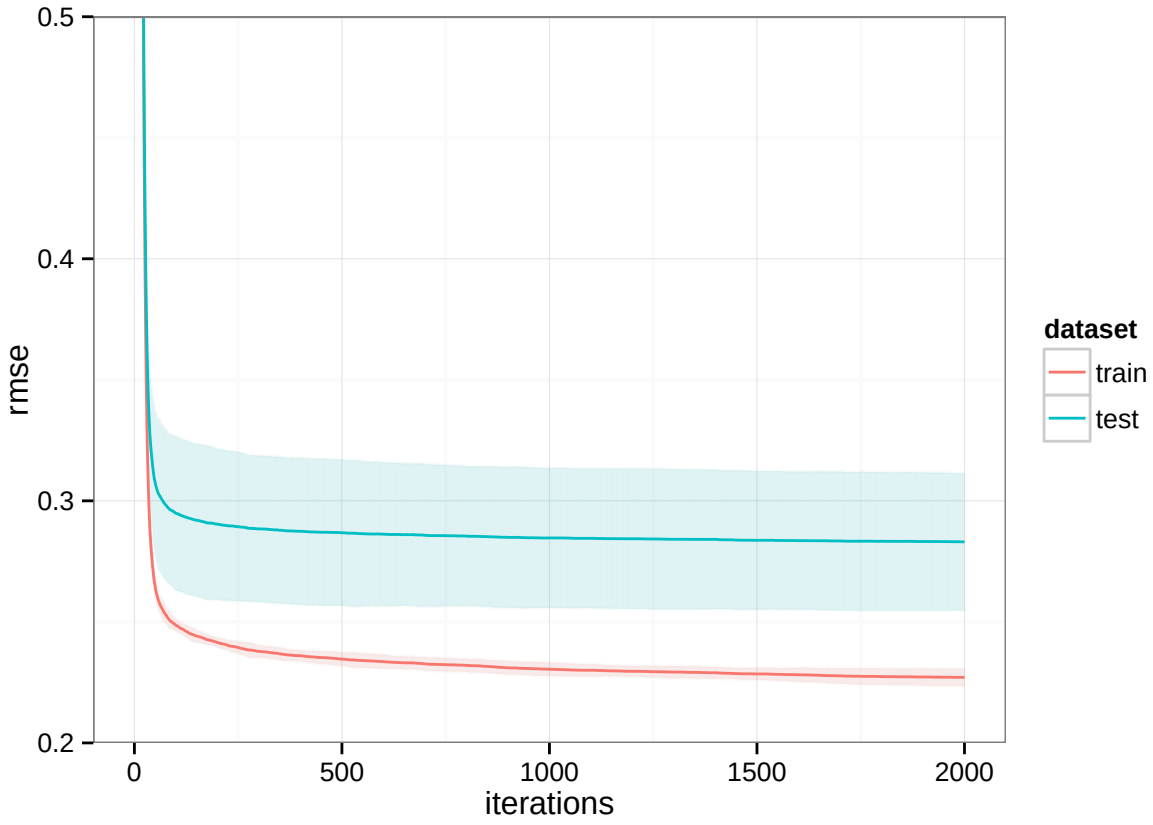
```
## [1] 0.3097785
```

Plot learning curve

```
plot_learning_curve = function(learning_curves) {
  learning_curves_train = NULL
  learning_curves_train$dataset = 'train'
  learning_curves_train$rmse = learning_curves$train.rmse.mean
  learning_curves_train$rmse.se = learning_curves$train.rmse.std
  learning_curves_train$iterations = 1:nrow(learning_curves)
  learning_curves_test = NULL
  learning_curves_test$dataset = 'test'
  learning_curves_test$rmse = learning_curves$test.rmse.mean
  learning_curves_test$rmse.se = learning_curves$test.rmse.std
  learning_curves_test$iterations = 1:nrow(learning_curves)
  learning_curves_prepared = rbind(as.data.frame(learning_curves_train),
                                   as.data.frame(learning_curves_test))
  ggplot(data = learning_curves_prepared,
         mapping = aes(x=iterations, y=rmse, group = dataset, colour=dataset)) +
    geom_errorbar(aes(ymin=rmse-2*rmse.se, ymax=rmse+2*rmse.se), width=.01, alpha=0.02) +
    geom_line() + coord_cartesian(ylim = c(0.2, 0.5))

}

plot_learning_curve(history)
```



Feature importance

```
imp = xgb.importance(colnames(train_matrix), model = model)
```

```
xgb.plot.importance(imp)
```


Feature importance
hour
hour31
temp
season
workingday
year
humidity
workingday:hour61
workingday:hour32
casual_ratio1.month
hour82
hour41
hour121
workingday:hour42
weather
hour36
wday
avg_count.month
hour63
workingday:hour31
workingday:hour81
var_count.hour
atemp − temp
hour62
var_count.month
avg_count.hour
workingday:hour41
hour81
casual_ratio2.month
hour44
casual_ratio1.hour
casual_ratio2.hour
workingday:hour62
var_registered.hour
windspeed
workingday:hour121
hour37
workingday:hour122
hour33
workingday:hour43
hour61
workingday:hour82
hour42
workingday:hour45
hour35
var_registered.month
hour32
hour31:holiday
workingday:hour35
holiday
hour61:holiday
hour42:holiday
hour43
workingday:hour44
hour45
hour32:holiday
workingday:hour37
hour41:holiday
hour38
workingday:hour63
hour37:holiday
hour46:holiday
hour122:holiday
workingday:hour36
hour45:holiday
hour46
workingday:hour33
hour81:holiday
hour62:holiday
Features
0.0
0.2
0.4
0.6
Gain
Cluster
1
2
3
4
5
6

Train on full dataset

```
dtrain_final <- xgb.DMatrix(full_train_matrix, label = preparedFullTarget)

model_final = xgboost(params = xgbControl, data = dtrain_final,
                      nround=2000, nthread = 4, print.every.n = 500)
```

```
## [0]  train-rmse:3.863858
## [500]    train-rmse:0.233306
## [1000]   train-rmse:0.228246
## [1500]   train-rmse:0.226401
```

```
history <- xgb.cv(params = xgbControl, data = dtrain_final,
                  nround=2000, nthread = 4, print.every.n = 500, nfold = 3)
```

```
## [0]  train-rmse:3.863931+0.004294    test-rmse:3.864195+0.009682
## [500]    train-rmse:0.238692+0.000752    test-rmse:0.287640+0.006449
## [1000]   train-rmse:0.234104+0.000233    test-rmse:0.285751+0.006612
## [1500]   train-rmse:0.231935+0.000594    test-rmse:0.285175+0.006573
```

Predict and un-log predictions

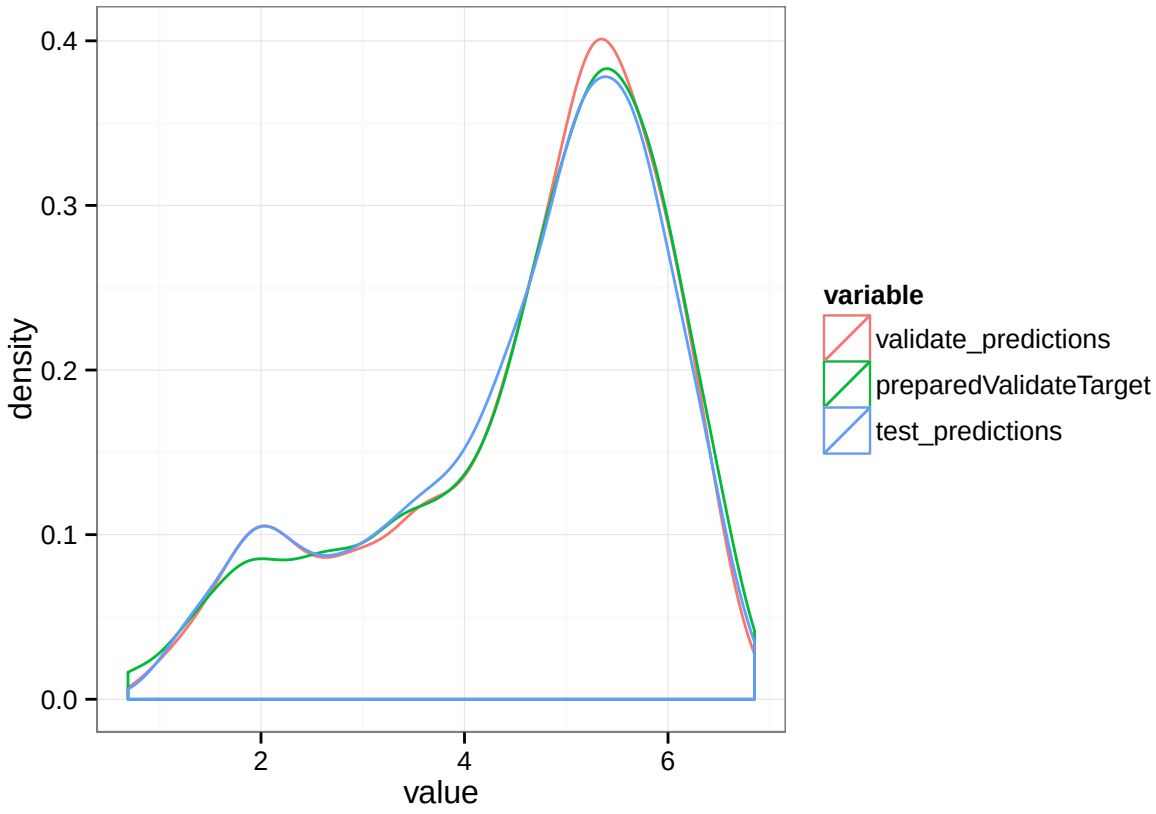
```
predictions = predict(model_final, test_matrix)
fixed_predictions = exp(predictions) - 1
```

Prediction vs Target density plot

```
test_predictions = sample(predictions, length(validate_predictions))
count_value = data.frame(validate_predictions, preparedValidateTarget, test_predictions)
count_value = melt(count_value)
```

```
## No id variables; using all as measure variables
```

```
ggplot(count_value, aes(group = variable, color = variable, x = value)) + geom_density()
```



Write the result

```
result = cbind(as.character(test$datetime), fixed_predictions) %>% as.data.frame()
names(result) = c('datetime', 'count')
write.csv(result, 'submission.csv', quote = F, row.names = F)
```