Bike sharing

Load default libraries

```
library(data.table)
library(dplyr)
library(lubridate)
library(ggplot2)
library(caret)
library(stringi)
library(xgboost)
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 = as.character(workingday),
      weather = as.character(weather)) %>%
    mutate(
      year = as.character(year(datetime)),
      month = as.character(month(datetime)),
      day = as.character(mday(datetime)),
      wday = as.character(wday(datetime)),
      hour = as.character(hour(datetime)))

}
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=0.5, colsample_bytree = 0.7, metrics=list("rmse"), min_child_weight = 30,
                  max.depth = 6, eta = 0.1, alpha = 2, lambda = 20, objective = "reg:linear"
)
model = xgboost(params = xgbControl, data = dtrain,
                nround=200, nthread = 2, print.every.n = 33)
```

```
## [0]  train-rmse:3.911107
## [33] train-rmse:0.644221
## [66] train-rmse:0.518934
## [99] train-rmse:0.468903
## [132]    train-rmse:0.432300
## [165]    train-rmse:0.408868
## [198]    train-rmse:0.392686
```

```
history <- xgb.cv(params = xgbControl, data = dtrain,
                  nround=200, nthread = 2, nfold = 5, print.every.n = 33)
```

```
## [0]  train-rmse:3.912580+0.014840    test-rmse:3.911532+0.050618
## [33] train-rmse:0.653399+0.009208    test-rmse:0.666574+0.022063
## [66] train-rmse:0.526634+0.009450    test-rmse:0.545976+0.024810
## [99] train-rmse:0.475363+0.007436    test-rmse:0.500728+0.027595
## [132]    train-rmse:0.438607+0.007621    test-rmse:0.468307+0.019740
## [165]    train-rmse:0.417607+0.005123    test-rmse:0.452919+0.021073
## [198]    train-rmse:0.398390+0.004472    test-rmse:0.438666+0.019818
```

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

```
##    train.rmse.mean train.rmse.std test.rmse.mean test.rmse.std
## 1:        0.400048       0.004286       0.439942      0.019769
## 2:        0.399669       0.004382       0.439680      0.019886
## 3:        0.399301       0.004361       0.439381      0.020064
## 4:        0.398964       0.004395       0.439143      0.020019
## 5:        0.398390       0.004472       0.438666      0.019818
## 6:        0.397714       0.004006       0.438031      0.020059
```

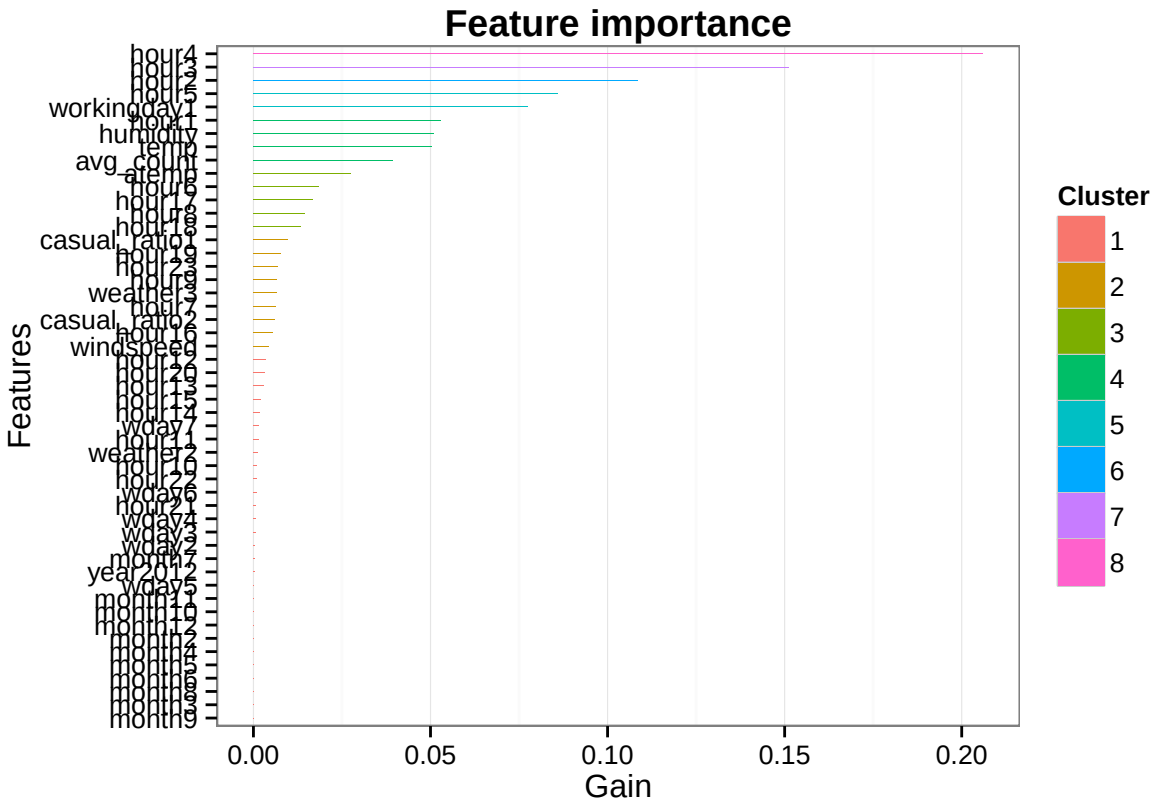
Validate Score

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

```
## [1] 0.4382262
```

Feature importance

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



Train on full dataset

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

model_final = xgboost(params = xgbControl, data = dtrain_final,
                      nround=200, nthread = 2, print.every.n = 33)
```

```
## [0]  train-rmse:3.931569
## [33] train-rmse:0.613105
## [66] train-rmse:0.484068
## [99] train-rmse:0.436825
## [132]    train-rmse:0.402976
## [165]    train-rmse:0.386157
## [198]    train-rmse:0.369093
```

```
history <- xgb.cv(params = xgbControl, data = dtrain_final,
                  nround=200, nthread = 2, print.every.n = 33, nfold = 5)
```

```
## [0]  train-rmse:3.930991+0.009217    test-rmse:3.930983+0.022028
## [33] train-rmse:0.628282+0.012589    test-rmse:0.636427+0.014100
## [66] train-rmse:0.507599+0.013540    test-rmse:0.521487+0.014730
## [99] train-rmse:0.448378+0.006443    test-rmse:0.467431+0.023430
## [132]    train-rmse:0.417753+0.003873    test-rmse:0.440241+0.022149
## [165]    train-rmse:0.395348+0.004240    test-rmse:0.421052+0.022239
## [198]    train-rmse:0.377776+0.002749    test-rmse:0.406586+0.019885
```

Predict and un-log predictions

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

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)
```