Bike sharing

Load default libraries

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

Read the data

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

Change type to categorical variables

```
train = train %>% mutate(
  season = as.character(season),
  holiday = as.character(holiday),
  workingday = as.character(workingday),
  weather = as.character(weather)
)

test = test %>% mutate(
  season = as.character(season),
  holiday = as.character(holiday),
  workingday = as.character(workingday),
  weather = as.character(weather)
)
```

Prepare features

Prepare target

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

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

Xgboost train and cross-validate

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

model = xgboost(dtrain, nround=100, nthread = 2, nfold = 5, metrics=list("rmse"),
                  max.depth =3, eta = 0.1, objective = "reg:linear", print.every.n = 25)
```

```
## [0]  train-rmse:3.935025
## [25] train-rmse:1.212616
## [50] train-rmse:1.160944
## [75] train-rmse:1.151518
```

```
history <- xgb.cv(data = dtrain, nround=100, nthread = 2, nfold = 5, metrics=list("rmse"),
                  max.depth =3, eta = 0.1, objective = "reg:linear", print.every.n = 25)
```

```
## [0]  train-rmse:3.935065+0.003580    test-rmse:3.935393+0.016221
## [25] train-rmse:1.210536+0.003635    test-rmse:1.222840+0.009228
## [50] train-rmse:1.157760+0.003588    test-rmse:1.176131+0.014401
## [75] train-rmse:1.148643+0.003925    test-rmse:1.171448+0.015027
```

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

```
##    train.rmse.mean train.rmse.std test.rmse.mean test.rmse.std
## 1:        1.143989       0.004164       1.169443      0.015506
## 2:        1.143812       0.004122       1.169376      0.015663
## 3:        1.143549       0.004271       1.169192      0.015630
## 4:        1.143385       0.004336       1.169092      0.015532
## 5:        1.143094       0.004352       1.168948      0.015517
## 6:        1.142948       0.004414       1.168865      0.015498
```

Predict and un-log count

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

Write the result

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