Logistic Regression

Xingche Guo

4/13/2019

Preprocess

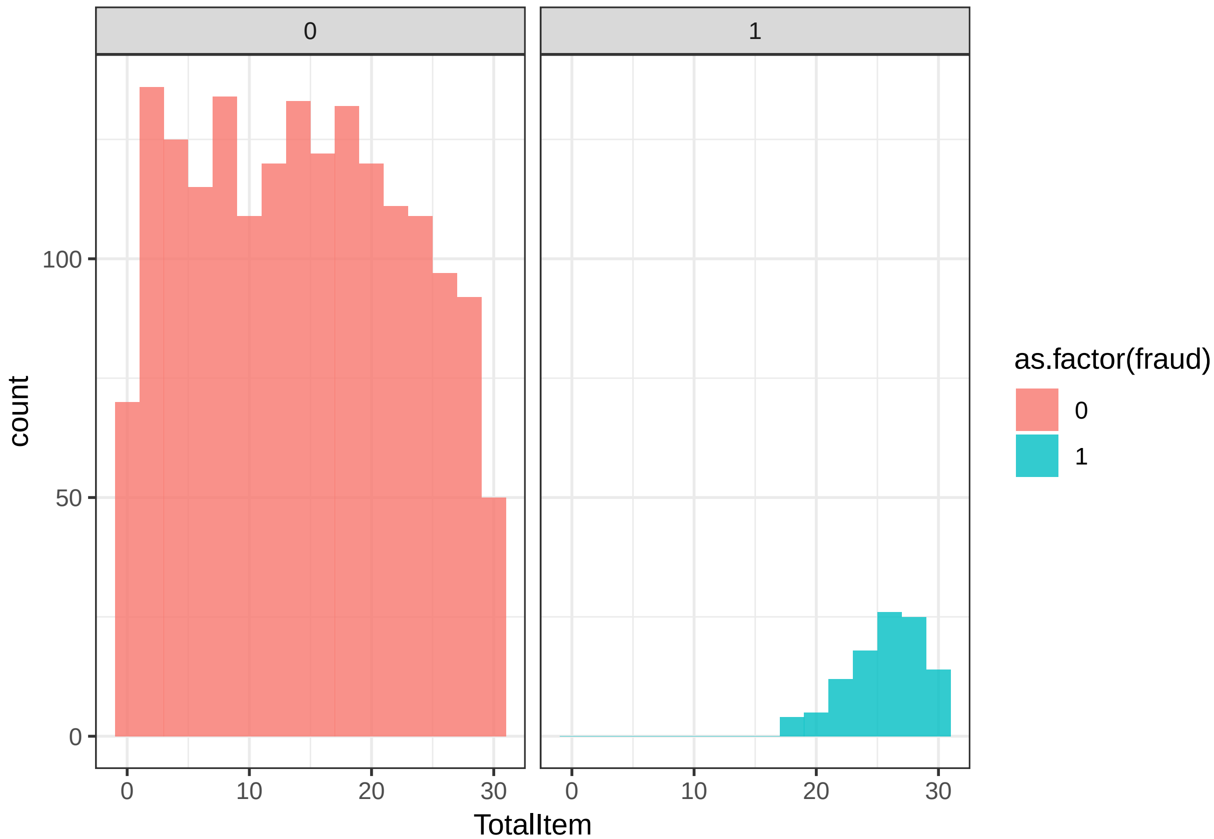
```
library(ggplot2)

Train <- read.csv(file = "/Users/apple/Desktop/ISU 2019 spring/DMC2019/DMC_2019_task/train.csv",
                  sep = "|")

names(Train)
```

```
##  [1] "trustLevel"                "totalScanTimeInSeconds"
##  [3] "grandTotal"                "lineItemVoids"
##  [5] "scansWithoutRegistration"  "quantityModifications"
##  [7] "scannedLineItemsPerSecond" "valuePerSecond"
##  [9] "lineItemVoidsPerPosition"  "fraud"
```

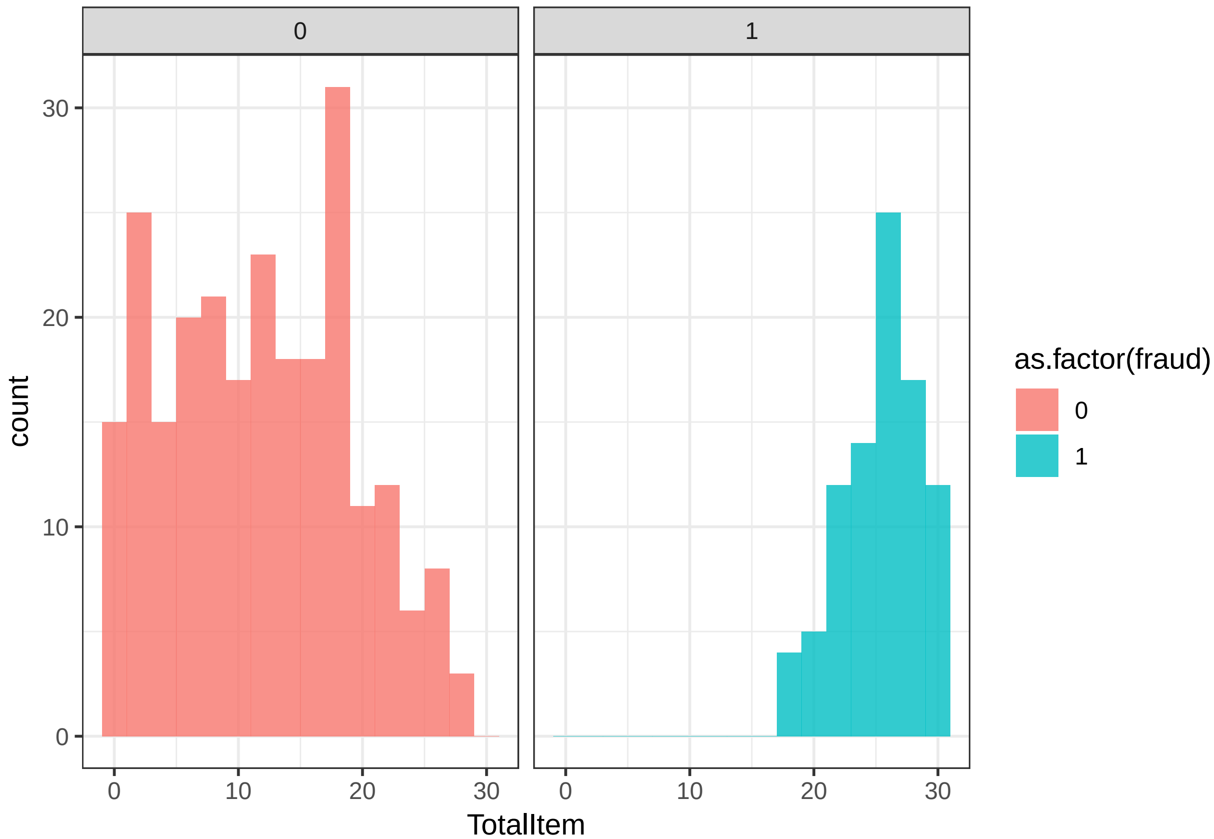
```
Train$trustLevel <- as.factor(Train$trustLevel)
TotalItem <- Train$totalScanTimeInSeconds * Train$scannedLineItemsPerSecond
AveValue <- Train$grandTotal / TotalItem

Train1 <- data.frame(Train, TotalItem = TotalItem)

level1_ind <- which(as.numeric(Train1$trustLevel)==1)
level2_ind <- which(as.numeric(Train1$trustLevel)==2)
```

Plot total item vs fraud

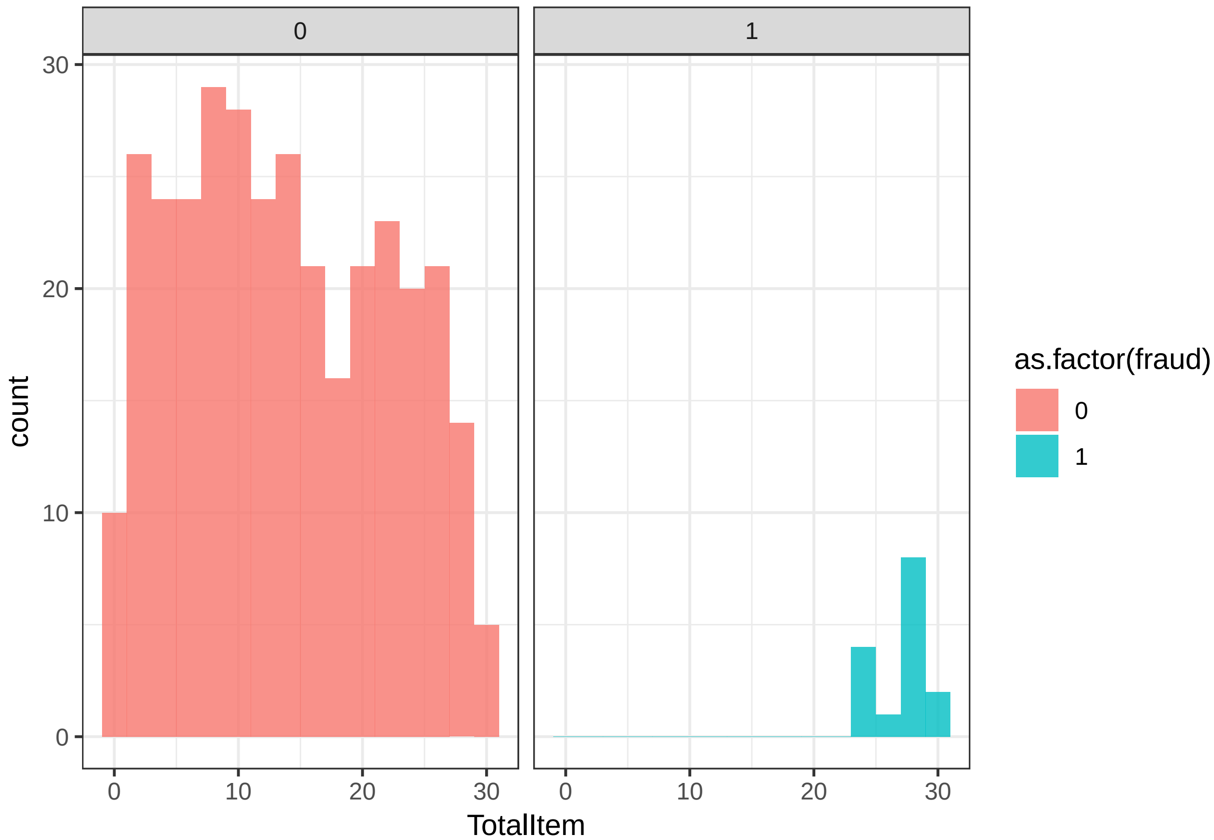
```
## all
ggplot(data = Train1) +
  geom_histogram(aes(x = TotalItem, fill = as.factor(fraud)), binwidth = 2, alpha = 0.8) +
  facet_grid(~as.factor(fraud)) +
  theme_bw()
```




```
## level1
ggplot(data = Train1[level1_ind,]) +
  geom_histogram(aes(x = TotalItem, fill = as.factor(fraud)), binwidth = 2, alpha = 0.8) +
  facet_grid(~as.factor(fraud)) +
  theme_bw()
```




```r
## level2
ggplot(data = Train1[level2_ind,]) +
  geom_histogram(aes(x = TotalItem, fill = as.factor(fraud)), binwidth = 2, alpha = 0.8) +
  facet_grid(~as.factor(fraud)) +
  theme_bw()
```



CV & logistic functions

```
################################### logistic regression ##################################
lossDMC <- function(true, pred){
  loss <- sum( (true==1) & (pred==0) ) * 5 +
    sum( (true==0) & (pred==1) ) * 25 +
    sum( (true==1) & (pred==1) ) * (-5)
  return(loss)
}





cv_design <- function(n, fold = 10){
  m <- floor(n/fold)
  r <- n%%fold
  p1 <- rep(m, fold)
  p2 <- rep(0, fold)
  if (r>=1){
    p2[1:r] <- 1
  }
  p <- p1 + p2
  ub <- cumsum(p)
  lb <- ub - p + 1
```

```
  x <- sample(n)
  IND <- vector("list",fold)
  for (i in 1:fold){
    IND[[i]] <- x[(lb[i]):(ub[i])]
  }
  return(IND)
}
```

```
cv_logistic_probs <- function(D,formula, fold = 10){
  n <- length(D[,1])
  IND <- cv_design(n, fold)
  probs <- rep(0,n)
  options(warn=-1)
  for (i in 1:fold){
    test_ind <- IND[[i]]
    Train <- D[-test_ind,]
    Test <- D[test_ind,]
    fit <- glm(formula=formula,data=Train,family=binomial(link=logit))
    py <- predict(fit,newdata=data.frame(Test), type = "response")
    probs[test_ind] <- py
  }
  return(probs)
}
```

Repeated CV results (100 repeated, 10 fold, Cross-Validation)

all trained as 0 (not fraud)

```
pred0 <- rep(0, length(Train[,1]))

## DMC loss
lossDMC(true = Train1$fraud,
        pred = pred0 )
```

```
## [1] 520
```

```
## 0-1 loss
sum(pred0!=Train1$fraud)
```

```
## [1] 104
```

Use total item as covariate

```
formula1 <- "fraud~."
loss1 <- rep(0, 100)
loss2 <- rep(0, 100)
for (i in 1:100){
  pred1 <- pred0
```

```r
  probs <- cv_logistic_probs(Train1, formula1, 10)
  pred1[ which(  probs > 5/7  ) ] <- 1
  loss1[i] <- lossDMC(true = Train1$fraud,
                      pred = pred1)
  loss2[i] <- sum(pred1!=Train1$fraud)
}

## DMC loss
mean(loss1)
```

```
## [1] -257.15
```

```r
## 0-1 loss
mean(loss2)
```

```
## [1] 16.01
```

Not use total item as covariate

```r
formula1 <- "fraud~."
loss1 <- rep(0, 100)
loss2 <- rep(0, 100)
for (i in 1:100){
  pred1 <- pred0
  probs <- cv_logistic_probs(Train, formula1, 10)
  pred1[ which(  probs > 5/7  ) ] <- 1
  loss1[i] <- lossDMC(true = Train$fraud,
                      pred = pred1)
  loss2[i] <- sum(pred1!=Train$fraud)
}

## DMC loss
mean(loss1)
```

```
## [1] -93.1
```

```r
## 0-1 loss
mean(loss2)
```

```
## [1] 32.49
```

Fit all the data

```r
fit_final <- glm(fraud~., data=Train1, family=binomial(link=logit))
summary(fit_final)
```

```
##
## Call:
## glm(formula = fraud ~ ., family = binomial(link = logit), data = Train1)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.271   0.000   0.000   0.000   1.627
```

```
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -1.134e+02  3.243e+01  -3.498 0.000469 ***
## trustLevel2                -2.296e+01  6.201e+00  -3.703 0.000213 ***
## trustLevel3                -6.776e+01  5.892e+03  -0.012 0.990824
## trustLevel4                -5.106e+01  6.449e+03  -0.008 0.993683
## trustLevel5                -6.237e+01  6.294e+03  -0.010 0.992093
## trustLevel6                -6.521e+01  6.104e+03  -0.011 0.991477
## totalScanTimeInSeconds      1.069e-02  3.205e-03   3.335 0.000854 ***
## grandTotal                  1.201e-01  4.595e-02   2.613 0.008962 **
## lineItemVoids               1.348e+00  7.452e-01   1.809 0.070464 .
## scansWithoutRegistration    2.137e+00  6.050e-01   3.531 0.000413 ***
## quantityModifications       2.077e-01  2.988e-01   0.695 0.487042
## scannedLineItemsPerSecond   1.644e+01  9.020e+00   1.823 0.068321 .
## valuePerSecond             -7.929e+01  2.795e+01  -2.837 0.004553 **
## lineItemVoidsPerPosition    5.630e+00  1.575e+01   0.357 0.720798
## TotalItem                   3.543e+00  1.026e+00   3.454 0.000552 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 804.108  on 1878  degrees of freedom
## Residual deviance:  32.173  on 1864  degrees of freedom
## AIC: 62.173
##
## Number of Fisher Scoring iterations: 24
```

```r
pred_final <- pred0
pred_final[which(fit_final$fitted.values>5/7)] <- 1

## DMC loss
lossDMC(true = Train$fraud,
        pred = pred_final)
```

```
## [1] -385
```

```r
## 0-1 loss
sum(pred_final!=Train1$fraud)
```

```
## [1] 9
```