Sampling

Karen Mazidi

Using a UCI data base: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients#

Load data

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.3
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
default_full <- read.csv("data/default.csv", header=TRUE)
default_full$default <- factor(default_full$default)
summary(default_full$default)
```

```
##     0     1
## 23364  6636
```

```
default_rate = nrow(default_full[default_full$default==1,]) / nrow(default_full)
print(paste("default rate = ", default_rate))
```

```
## [1] "default rate =  0.2212"
```

```
# limit columns
default_full <- default_full[,c(2:8,13,19,20,25)]
```

Train and test

```
set.seed(1234)
i <- sample(1:nrow(default_full), 0.8*nrow(default_full), replace=FALSE)
train_full <- default_full[i,]
test <- default_full[-i,]
```

Logistic Regression with all rows

80%

```
glm1 <- glm(default~., data=train_full, family=binomial)
probs <- predict(glm1, newdata=test, type='response')
pred <- ifelse(probs>0.5, 1, 0)
confusionMatrix(pred, test$default)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 4483 1078
##          1  121  318
```

```
##
##                Accuracy : 0.8002
##                  95% CI : (0.7898, 0.8102)
##     No Information Rate : 0.7673
##     P-Value [Acc > NIR] : 5.086e-10
##
##                   Kappa : 0.2647
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9737
##             Specificity : 0.2278
##          Pos Pred Value : 0.8061
##          Neg Pred Value : 0.7244
##              Prevalence : 0.7673
##          Detection Rate : 0.7472
##    Detection Prevalence : 0.9268
##       Balanced Accuracy : 0.6008
##
##        'Positive' Class : 0
##
```

```r
mean(pred==test$default)
```

```
## [1] 0.8001667
```

Reduce data set to 1000

```r
set.seed(1234)
j <- sample(1:30000, 1000, replace=FALSE)
df_1000_random <- default_full[j,]
```

Logistic regression on random sample

84%

```r
set.seed(1234)
i <- sample(1:1000, 800, replace=FALSE)
train <- df_1000_random[i,]
test <- df_1000_random[-i,]
glm2 <- glm(default~., data=train, family=binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
probs <- predict(glm2, newdata=test, type='response')
pred <- ifelse(probs>0.5, 1, 0)
confusionMatrix(pred, test$default)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 154  25
##          1   7  14
##
```

```
##                Accuracy : 0.84
##                  95% CI : (0.7817, 0.8879)
##     No Information Rate : 0.805
##     P-Value [Acc > NIR] : 0.121512
##
##                   Kappa : 0.3824
##  Mcnemar's Test P-Value : 0.002654
##
##             Sensitivity : 0.9565
##             Specificity : 0.3590
##          Pos Pred Value : 0.8603
##          Neg Pred Value : 0.6667
##              Prevalence : 0.8050
##          Detection Rate : 0.7700
##    Detection Prevalence : 0.8950
##       Balanced Accuracy : 0.6577
##
##        'Positive' Class : 0
##
```

```r
mean(pred==test$default)
```

```
## [1] 0.84
```

Stratified sampling with caret

The following code randomly samples 80% of the rows of the data set while preserving the distribution.

```r
k <- createDataPartition(default_full$default, p=0.8, list=FALSE)
```

Logistic regression on full data set with startified sampline

```r
train_k <- default_full[k,]
test_k <- default_full[-k,]

default_rate_k = nrow(train_k[train_k$default==1,]) / nrow(train_k)
#same


glm3 <- glm(default~., data=train_k, family=binomial)
probs <- predict(glm3, newdata=test_k, type='response')
pred <- ifelse(probs>0.5, 1, 0)
confusionMatrix(pred, test_k$default)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 4518 1020
##          1  154  307
##
##                Accuracy : 0.8043
##                  95% CI : (0.794, 0.8143)
```

```
##     No Information Rate : 0.7788
##     P-Value [Acc > NIR] : 7.555e-07
##
##                   Kappa : 0.2589
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9670
##             Specificity : 0.2313
##          Pos Pred Value : 0.8158
##          Neg Pred Value : 0.6659
##              Prevalence : 0.7788
##          Detection Rate : 0.7531
##    Detection Prevalence : 0.9232
##       Balanced Accuracy : 0.5992
##
##        'Positive' Class : 0
##
```

```r
mean(pred==test_k$default)
```

```
## [1] 0.8043007
```

Make a more balanced data set

We are going to make a train set that is smaller than the train set above but is more evenly distributed to see how that impacts the algorithm.

As seen below, having 50% of each class lead to dramatically worse performance by logistic regression.

```r
train_0 <- train_k[which(train_k$default==0),]
train_1 <- train_k[which(train_k$default==1),]
set.seed(1234)
j <- sample(1:5300, 5000, replace=FALSE)
train_bal <- rbind(train_0[j,], train_1[1:5000,])

glm4 <- glm(default~., data=train_bal, family=binomial)
probs <- predict(glm4, newdata=test_k, type='response')
pred <- ifelse(probs>0.5, 1, 0)
confusionMatrix(pred, test_k$default)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3466  543
##          1 1206  784
##
##                Accuracy : 0.7085
##                  95% CI : (0.6968, 0.7199)
##     No Information Rate : 0.7788
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2822
##  Mcnemar's Test P-Value : <2e-16
##
```

```
##             Sensitivity : 0.7419
##             Specificity : 0.5908
##          Pos Pred Value : 0.8646
##          Neg Pred Value : 0.3940
##              Prevalence : 0.7788
##          Detection Rate : 0.5778
##    Detection Prevalence : 0.6683
##       Balanced Accuracy : 0.6663
##
##        'Positive' Class : 0
##
```

```r
mean(pred==test_k$default)
```

```
## [1] 0.7084514
```