

caiomsouza / **kaggle-competitions**

Unwatch

1

Star

0

Fork

0

Code

Issues 0

Pull requests 0

Wiki

Pulse

Graphs

Settings

Branch: master


kaggle-competitions / santander-customer-satisfaction /

Create new file

Upload files

Find file

History

 caiomsouza Last commit before sending to Javier Portela	Latest commit 93141d4 17 seconds ago
..	
dat	Add Kaggle Santander Competition 3 days ago
extras	add 8 hours ago
images/prueba_grid_4.R_images	add images 41 minutes ago
outputs	add 4 hours ago
src	Last commit before sending to Javier Portela 4 minutes ago
README.md	Last commit before sending to Javier Portela 17 seconds ago
kaggle-competitions:santander-customer-satisfaction ...	add 8 hours ago

README.md

1. Student and Professor

Student:
Caio Fernandes Moreno (caiofern@ucm.es)
twitter: @caiomsouza

Professor:
Prof. Phd. Javier Portella

The Project:
All the files for the class project is available at:
<https://github.com/caiomsouza/kaggle-competitions/tree/master/santander-customer-satisfaction>

This is an open source project.
The MIT License (MIT)
<https://opensource.org/licenses/MIT>

2. Faculty of Statistics at UCM

Prof. Phd. Javier Portella (Javier Portela javipgm@gmail.com)
UCM - Universidad Complutense de Madrid
Facultad de Estudios Estadísticos
Avda. Puerta de Hierro s/n
Ciudad Universitaria
28040-MADRID

3. Kaggle Santander Customer Satisfaction Competition

This project was created by Caio Fernandes Moreno, student at the faculty of statistics at UCM - Complutense University of Madrid - Universidad Complutense de Madrid.
The main purpose of this academic project is to learn data mining techniques, statistics, machine learning, R, SAS, data preparation, etc.

Neuronal Networks and Genetic Algorithms / Redes Neuronales y Algoritmos Genéticos
Master degree in Data Mining and Business Intelligence

Kaggle Competition dates:

Started: 7:43 pm, Wednesday 2 March 2016 UTC

Ended: 11:59 pm, Monday 2 May 2016 UTC (61 total days)

Competition link:

<https://www.kaggle.com/c/santander-customer-satisfaction>

4. The problem

Which customers are happy customers?

From frontline support teams to C-suites, customer satisfaction is a key measure of success. Unhappy customers don't stick around. What's more, unhappy customers rarely voice their dissatisfaction before leaving.

Santander Bank is asking Kagglers to help them identify dissatisfied customers early in their relationship. Doing so would allow Santander to take proactive steps to improve a customer's happiness before it's too late.

In this competition, you'll work with hundreds of anonymized features to predict if a customer is satisfied or dissatisfied with their banking experience.

5. Evaluation and Submission File

Evaluation

Submissions are evaluated on area under the ROC curve between the predicted probability and the observed target.

Submission File

For each ID in the test set, you must predict a probability for the TARGET variable. The file should contain a header and have the following format:

```
ID, TARGET
2, 0
5, 0
6, 0
etc.
```

6. Dataset

The dataset provided by Santander Bank is anonymized and contains 371 variables (all continuous variables).

A continuous variable is a variable that has an infinite number of possible values. In other words, any value is possible for the variable. A continuous variable is the opposite of a discrete variable, which can only take on a certain number of values. Sep 11, 2013

The TARGET column is the variable to predict. It equals 1 (one) for unsatisfied customers and 0 for satisfied customers.

The Kaggle Competition Objective is to predict who are satisfied and unsatisfied clients.

Numbers of observations (Row number):

- Train: 76020 rows
- Test: 75818 rows

Number of 1s (train): 3008 (3.95%) (Imbalanced Dataset Problem)

Variables:

- 34 variables with one single value; (Action: Delete all of them)
- 100 variables with two unique values; (binary variables)
- 157 variables with values between 3 y 101 unique values; (categorical variables)
- 80 variables has more than 101 distinct values; (continuous variables)

Files:

- train.csv - with TARGET variable

- test.csv - without TARGET variable

Files at: <https://github.com/caiomsouza/kaggle-competitions/tree/master/santander-customer-satisfaction/dat>

7. Unbalanced Dataset Problem

A dataset is said to be unbalanced when the class of interest (minority class) is much rarer than normal behaviour (majority class). The cost of missing a minority class is typically much higher than missing a majority class. Most learning systems are not prepared to cope with unbalanced data and several techniques have been proposed to rebalance the classes. This package implements some of most well-known techniques and propose a racing algorithm [2] to select adaptively the most appropriate strategy for a given unbalanced task [2].

Some links about Unbalanced Dataset Problem:

- <http://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/>
- <http://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf>
- <http://florianhartl.com/thoughts-on-machine-learning-dealing-with-skewed-classes.html>
- <https://cran.r-project.org/web/packages/unbalanced/unbalanced.pdf>
- <https://www3.nd.edu/~dial/publications/chawla2005data.pdf>

8. ML Algorithms

Below is the list of the algorithms I must use in this project:

- Neuronal Networks (Redes Neuronales)
- Logistic Regression (Regresión Logística)
- Random Forest
- Bagging
- Gradient Boosting
- Support Vector Machines (try at least polynomial kernel with grade 1 - kernel polinómico de grado 1)
- Simple decision Tree to understand the dataset;

8.1 Neuronal Networks (Redes Neuronales)

8.2 Logistic Regression (Regresión Logística)

8.3 Random Forest

8.4 Bagging

8.5 Gradient Boosting Machine

8.6 Support Vector Machines

8.7 Simple Decision Tree

9. Load data

```
../src/utis/load_datasets.R
```

```
rm(list = ls())
setwd("~/git/github.com/kaggle-competitions/santander-customer-satisfaction")
path <- setwd("~/git/github.com/kaggle-competitions/santander-customer-satisfaction")

train.dataset.path <- "~/git/github.com/kaggle-competitions/santander-customer-satisfaction/dat/train.csv"
test.dataset.path <- "~/git/github.com/kaggle-competitions/santander-customer-satisfaction/dat/test.csv"

train <- read.csv(train.dataset.path)
cat("Train dataset loaded")
cat("at ")
cat(train.dataset.path)
```

```
test <- read.csv(test.dataset.path)
cat("Test dataset loaded")
cat("at ")
cat(test.dataset.path)

#head(train,5)
#head(test,5)
#nrow(train)
#nrow(test)
#colnames(train)
#colnames(test)
#summary(train)
#summary(test)
```

Code:

https://github.com/caiomsouza/kaggle-competitions/blob/master/santander-customer-satisfaction/src/utils/load_datasets.R

10. Data preparation

The script below will do data preparation for the Santander Customer Satisfaction dataset.

- Extracting TARGET
- 0 count per line
- Removing constant features
- Removing identical features

../src/utils/clean_datasets.R

```
##### Removing IDs
#train$ID <- NULL
#test.id <- test$ID
#test$ID <- NULL

##### Extracting TARGET
train.y <- train$TARGET
train$TARGET <- NULL

##### 0 count per line
count0 <- function(x) {
  return( sum(x == 0) )
}
train$n0 <- apply(train, 1, FUN=count0)
test$n0 <- apply(test, 1, FUN=count0)

##### Removing constant features
cat("\n## Removing constant features.\n")
for (f in names(train)) {
  if (length(unique(train[[f]])) == 1) {
    cat("-", f, "\n")
    train[[f]] <- NULL
    test[[f]] <- NULL
  }
}

##### Removing identical features
cat("\n## Removing identical features.\n")
features_pair <- combn(names(train), 2, simplify = F)
toRemove <- c()
for(pair in features_pair) {
  f1 <- pair[1]
  f2 <- pair[2]

  if (!(f1 %in% toRemove) & !(f2 %in% toRemove)) {
    if (all(train[[f1]] == train[[f2]])) {
      cat("-", f2, "\n")
      toRemove <- c(toRemove, f2)
    }
  }
}

feature.names <- setdiff(names(train), toRemove)
```

```
train <- train[, feature.names]
test <- test[, feature.names]

train$TARGET <- train.y
```

Code:

https://github.com/caiomsouza/kaggle-competitions/blob/master/santander-customer-satisfaction/src/utlis/clean_datasets.R

Data preparation output:

```
## Removing constant features.
- ind_var2_0
- ind_var2
- ind_var27_0
- ind_var28_0
- ind_var28
- ind_var27
- ind_var41
- ind_var46_0
- ind_var46
- num_var27_0
- num_var28_0
- num_var28
- num_var27
- num_var41
- num_var46_0
- num_var46
- saldo_var28
- saldo_var27
- saldo_var41
- saldo_var46
- imp_amort_var18_hace3
- imp_amort_var34_hace3
- imp_reemb_var13_hace3
- imp_reemb_var33_hace3
- imp_trasp_var17_out_hace3
- imp_trasp_var33_out_hace3
- num_var2_0_ult1
- num_var2_ult1
- num_reemb_var13_hace3
- num_reemb_var33_hace3
- num_trasp_var17_out_hace3
- num_trasp_var33_out_hace3
- saldo_var2_ult1
- saldo_medio_var13_medio_hace3

## Removing identical features.
- ind_var29_0
- ind_var29
- ind_var13_medio
- ind_var18
- ind_var26
- ind_var25
- ind_var32
- ind_var34
- ind_var37
- ind_var39
- num_var29_0
- num_var29
- num_var13_medio
- num_var18
- num_var26
- num_var25
- num_var32
- num_var34
- num_var37
- num_var39
- saldo_var29
- saldo_medio_var13_medio_ult1
- delta_num_reemb_var13_1y3
- delta_num_reemb_var17_1y3
- delta_num_reemb_var33_1y3
- delta_num_trasp_var17_in_1y3
```

- delta_num_trasp_var17_out_1y3
- delta_num_trasp_var33_in_1y3
- delta_num_trasp_var33_out_1y3

11. Models

Model Number	Model File Name	Type	AUC	Execution Time
1	xgb.model.1.R	xgboost is short for eXtreme Gradient Boosting	0.825281	Less than 5 minutes
2	xgb.model.2.R	xgboost is short for eXtreme Gradient Boosting	0.826584	Less than 5 minutes
3	h2o.glm.model.1.R	H2o GLM	0.8027165	Less than 5 minutes
4	h2o.glm.model.2.R	H2o GLM	0.8025738	Less than 5 minutes
5	h2o.glm.model.3.R	H2o GLM	0.8025738	Less than 5 minutes
6	h2o.randomforest.model.1.R	H2o Random Forest	0.8107832	Between 5 min to 30 min
7	h2o.randomforest.model.2.R	H2o Random Forest	0.8107832	Between 5 min to 30 min
8	h2o.randomforest.model.3.R	H2o Random Forest	0.7802334	Between 5 min to 30 min
9	h2o.randomforest.model.4.R	H2o Random Forest	0.8107832	Between 5 min to 30 min
10	h2o.deeplearning.model.1.R	H2o deeplearning	0.8018106	Between 5 min to 30 min
11	h2o.deeplearning.model.2.R	H2o deeplearning	0.7790996	Between 5 min to 30 min
12	h2o.deeplearning.model.3.R	H2o deeplearning	?	Between 5 min to 30 min
13	h2o.gbm.model.1.R	H2o GBM	0.8766553	Between 5 min to 30 min
14	h2o.ensembles.model1.R	H2o ensembles	??	Between 5 min to 30 min
15	h2o.stack_models.R	H2o Stack Model (GLM, GBM, Random Forest, Deeplearning) with all variables	??	Between 5 min to 30 min
16	svm_model.1.R	SVM (Support Vector Machines)	??	More than 36 hours

17	grid%20final%20todos.R	Mix of algorithms	??	More than 36 hours
----	------------------------	-------------------	----	--------------------

12. Conclusion

The best model is the number 13 using H2o GBM. The reasons are:

- Good prediction power;
- Extreme Fast
- Extreme easy to use;
- Because of H2o ML Framework it is easy to scale this algorithm with Big Data.

Algorithm using Random Forest are good options too, but they are slow.

The use of SVM (Support Vector Machine) with this dataset is almost impossible, it takes hours and sometime days to execute it.

Ensembles models are probable the best choice, but it is a little bit difficult to use it and a little bit slow.

In the future I will add more models in this test table and make sure there is no better options.

Because of my lack of skills, time, etc I was not able to do a extensive research about more possibilities of models and I am sure there are better options.

Please, feel free to fork this project and help me find more models. Do not forget to send me a pull request If you find better models with better results.

13. Project Resources

- Two semester of classes (SEMMA and Neuronal Networks);
- More than 100 hours of personal dedication;
- A lot of days dedicate to create models and submit at Kaggle;
- 33 submitted models to Kaggle Platform during the competition;
- A personal notebook with the configuration below:
 - MacBook Pro (Retina, 15-inch, Mid 2014)
 - OS X El Capitan 10.11 Beta
 - Processor: 2,2 GHz Intel Core i7
 - Memory: 16 GB 1600 MHz DDR3

14. Personal Final Results

My *final position* was 1599 from 5123 (0.3121218 or between the top 31%) with AUC of 0.825471. The first place had an AUC of 0.829072.

The difference was 0.003601. First place receive a prize of \$30,000.

```
> 0.829072 - 0.825471
[1] 0.003601
```

Link to Private Leaderboard - Santander Customer Satisfaction <https://www.kaggle.com/c/santander-customer-satisfaction/leaderboard>

Prizes

1st place - \$30,000

2nd place - \$20,000

3rd place - \$10,000

<https://www.kaggle.com/c/santander-customer-satisfaction/details/prizes>

15. References

1. H2o.ai - Gradient Boosting Machine. Available at: <http://www.h2o.ai/verticals/algos/gbm/>

2. Dal Pozzolo, Andrea, et al. "Racing for unbalanced methods selection." Intelligent Data Engineering and Automated Learning - IDEAL 2013. Springer Berlin Heidelberg, 2013. 24-31.

16. Tutorials

1. Anomaly Detection on MNIST with H2O Deep Learning. Available at: https://github.com/h2oai/h2o-training-book/blob/master/hands-on_training/anomaly_detection.md
2. A little H2O deeplearning experiment on the MNIST data set. Available at: <http://www.r-bloggers.com/a-little-h2o-deeplearning-experiment-on-the-mnist-data-set/>
3. H2O GBM Tuning Tutorial for R. Available at: <http://blog.h2o.ai/2016/06/h2o-gbm-tuning-tutorial-for-r/>
4. Ensembles: Stacking, Super Learner. Available at: <https://github.com/h2oai/h2o-tutorials/tree/master/tutorials/ensembles-stacking>
5. Arno Candel - Anomaly Detection and Feature Engineering. Available at: <https://www.youtube.com/watch?v=fUSbljByXak>
6. H2O Meetups and Presentations. Available at: <https://github.com/h2oai/h2o-meetups>
7. Training materials for H2O World. Available at: <https://github.com/h2oai/h2o-training-book> or <https://www.gitbook.com/book/h2o/h2o-world-2015-training/details>
8. subsemble. Available at: <https://github.com/ledell/subsemble>
9. H2O Tutorials. Available at: <http://learn.h2o.ai/content/>
10. Example lending_club_bad_loans_ensemble.R. Available at: https://github.com/h2oai/h2o-3/blob/master/h2o-r/ensemble/demos/lending_club_bad_loans_ensemble.R
11. H2O World 2015 Ensembles. Available at: https://github.com/h2oai/h2o-tutorials/blob/master/tutorials/ensembles-stacking/H2O_World_2015_Ensembles.pdf

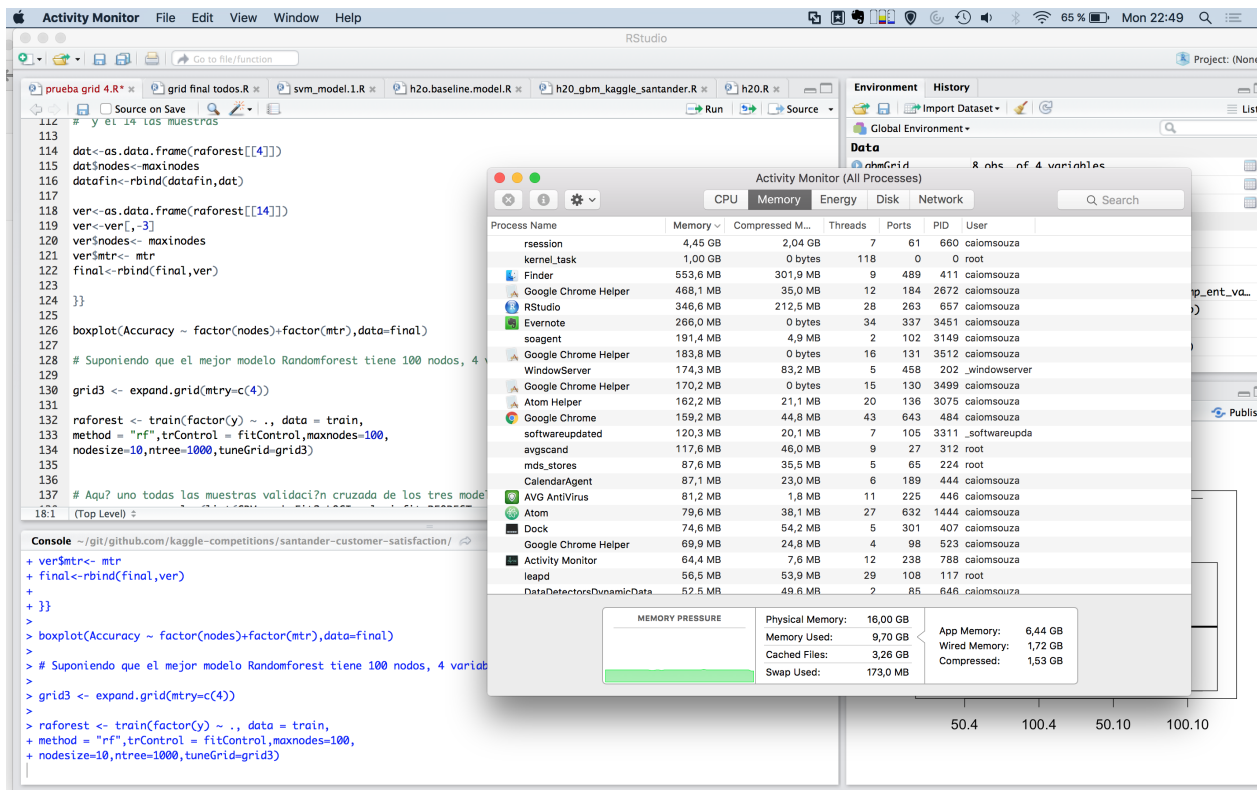
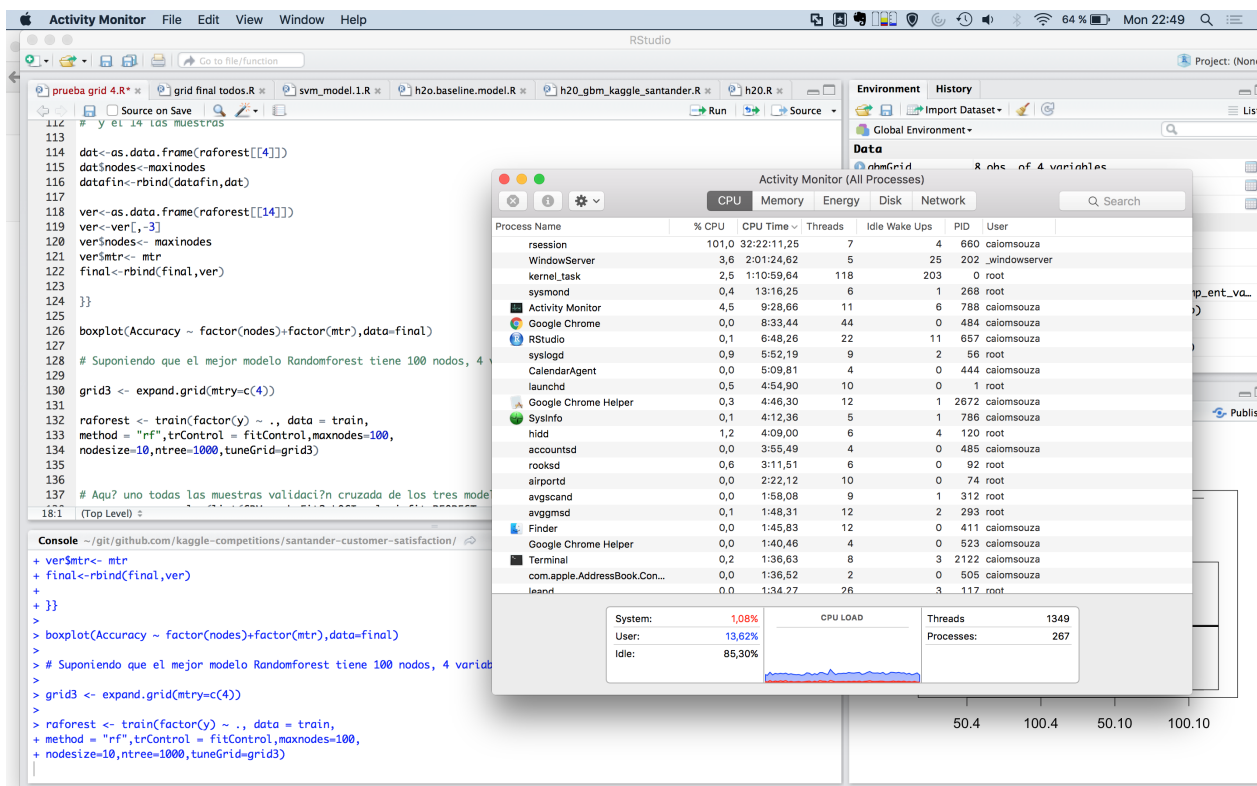
17. Links

<http://www.ats.ucla.edu/stat/r/dae/logit.htm>
<http://data.princeton.edu/R/linearModels.html>
<http://rischanlab.github.io/SVM.html>
<http://www.cs.upc.edu/~belanche/Docencia/mineria/English-september-2008/Practical-work/Labo-SVMs.R>
<http://www.svm-tutorial.com/2014/10/support-vector-regression-r/>
http://ethen8181.github.io/machine-learning/h2o/h2o_ensemble_tree/h2o_ensemble_tree.html
<https://github.com/h2oai/h2o-tutorials/tree/master/tutorials/ensembles-stacking>
https://github.com/h2oai/h2o-3/blob/master/h2o-r/ensemble/demos/h2o_ensemble_documentation_example.R
https://github.com/h2oai/h2o-3/blob/master/h2o-r/ensemble/demos/lending_club_bad_loans_ensemble.R

18. Super Machines

When I executed the script `prueba_grid_4.R` I was not able to wait the script to finish.

As we can see in the image below it took more than 32 hours to execute and I decided to stop it.



Before I stopped I've got some results:

```
library(caret)
Loading required package: lattice
Loading required package: ggplot2
Warning message:
package 'ggplot2' was built under R version 3.2.3
> library(gbm)
Loading required package: survival
```

Attaching package: 'survival'

The following object is masked from 'package:caret':

```
cluster
```

```

Loading required package: splines
Loading required package: parallel
Loaded gbm 2.1.1
> set.seed(1000)
> fitControl <- trainControl(method = "repeatedcv",number = 4,repeats = 10)
> y <- train$TARGET
> y <- train$TARGET
>
> set.seed(1000)
> fitControl <- trainControl(method = "repeatedcv",number = 4,repeats = 10)
>
> # Ejemplo grid para gbm
> gbmGrid <- expand.grid(interaction.depth = c(3,6),
+                         n.trees = c(100),
+                         shrinkage =c(0.1,0.02),
+                         n.minobsinnode = c(5,10))
>
> gbmFit1 <- train(factor(y) ~ ., data = train,
+                  method = "gbm",trControl = fitControl,
+                  verbose = FALSE,tuneGrid=gbmGrid)
Loading required package: plyr
There were 50 or more warnings (use warnings() to see the first 50)
> gbmFit1
Stochastic Gradient Boosting

```

```

76020 samples
308 predictor
2 classes: '0', '1'

```

No pre-processing

Resampling: Cross-Validated (4 fold, repeated 10 times)

Summary of sample sizes: 57015, 57015, 57015, 57015, 57015, 57015, ...

Resampling results across tuning parameters:

shrinkage	interaction.depth	n.minobsinnode	Accuracy	Kappa	Accuracy SD	Kappa SD
0.02	3	5	1	1	0	0
0.02	3	10	1	1	0	0
0.02	6	5	1	1	0	0
0.02	6	10	1	1	0	0
0.10	3	5	1	1	0	0
0.10	3	10	1	1	0	0
0.10	6	5	1	1	0	0
0.10	6	10	1	1	0	0

Tuning parameter 'n.trees' was held constant at a value of 100

Accuracy was used to select the optimal model using the largest value.

The final values used for the model were n.trees = 100, interaction.depth = 3, shrinkage = 0.02 and n.minobsinnode =

```

> gbmGrid <- expand.grid(interaction.depth = c(3),
+                         n.trees = c(100),shrinkage =c(0.02),
+                         n.minobsinnode = c(5))
>
> gbmFit2 <- train(factor(y) ~ ., data = train,
+                  method = "gbm",trControl = fitControl,
+                  verbose = FALSE,tuneGrid=gbmGrid)
There were 50 or more warnings (use warnings() to see the first 50)
>
>
> logisfit <- train(factor(y) ~ ., data = train,
+                  method = "glm",trControl = fitControl)
There were 50 or more warnings (use warnings() to see the first 50)
>
> logisfit
Generalized Linear Model

```

```

76020 samples
308 predictor
2 classes: '0', '1'

```

No pre-processing

Resampling: Cross-Validated (4 fold, repeated 10 times)

Summary of sample sizes: 57015, 57015, 57015, 57015, 57015, 57015, ...

Resampling results

Accuracy	Kappa	Accuracy SD	Kappa SD
0.9999961	0.9999481	1.403561e-05	0.0001845475

>

```

> # *****
> # EJEMPLO GRID RANDOM FOREST
> # (SOLO PERMITE VARIAR MTRY)
> # *****
>
> # Conservo el anterior fitControl de validaci?n cruzada para comparar
> # todos los modelos que voy creando
>
> grid2 <- expand.grid(mtry=c(4,8,10))
>
> # En esta ejecucion no pongo numero de muestra y utiliza
> # por defecto todas las observaciones CON reemplazo
>
> raforest <- train(factor(y) ~ ., data = train,
+ method = "rf",trControl = fitControl,maxnodes=50,
+ nodesize=10,ntree=100,tuneGrid=grid2)
Loading required package: randomForest
randomForest 4.6-10
Type rfNews() to see new features/changes/bug fixes.

Attaching package: 'randomForest'

The following object is masked from 'package:ggplot2':

    margin

>
> raforest
Random Forest

76020 samples
 308 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (4 fold, repeated 10 times)
Summary of sample sizes: 57015, 57015, 57015, 57015, 57015, ...
Resampling results across tuning parameters:

  mtry Accuracy   Kappa      Accuracy SD   Kappa SD
    4   0.9604315 0.0000000000 0.000000e+00 0.0000000000
    8   0.9604354 0.0001913411 1.403561e-05 0.0006805284
   10   0.9604512 0.0009567056 2.579797e-05 0.0012508360

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 10.

>
> # *****
> # Ejemplo Grid con par?metros que no est?n considerados en caret
> # Por ejemplo quiero variar maxnodes y mtry en random forest
> # *****
>
> # Creo datafin y final archivo vac?o
>
> datafin=data.frame(mtry=numeric(0),Accuracy=numeric(0),Kappa=numeric(0),
+ AccuracySD=numeric(0),KappaSD=numeric(0),nodes=numeric(0))
>
> final=data.frame(Accuracy=numeric(0),Kappa=numeric(0), nodes=numeric(0),mtr=numeric(0))
>
> # bucle para maxnodes y mtry
>
> for(maxinodes in c(50,100)) {
+
+   for(mtr in c(4,10)) {
+
+     grid3 <- expand.grid(mtry=c(mtr))
+
+     raforest <- train(factor(y) ~ ., data = train,
+ method = "rf",trControl = fitControl,maxnodes=maxinodes,
+ nodesize=10,ntree=200,tuneGrid=grid3)
+
+     # El elemento 4 de la lista raforest son los resultados globales
+     # y el 14 las muestras
+
+     dat<-as.data.frame(raforest[[4]])
+     dat$nodes<-maxinodes
+     datafin<-rbind(datafin,dat)
+
+   }
+ }

```

```
+ ver<-as.data.frame(raforest[[14]])
+ ver<-ver[, -3]
+ ver$nodes<- maxnodes
+ ver$mtr<- mtr
+ final<-rbind(final,ver)
+
+ }}
>
> boxplot(Accuracy ~ factor(nodes)+factor(mtr),data=final)
>
> # Suponiendo que el mejor modelo Randomforest tiene 100 nodos, 4 variables
>
> grid3 <- expand.grid(mtry=c(4))
>
> raforest <- train(factor(y) ~ ., data = train,
+ method = "rf",trControl = fitControl,maxnodes=100,
+ nodesize=10,ntree=1000,tuneGrid=grid3)
```

