HarvardX. Module 9: Data Science
Final Project: E-commerce conversion prediction

Adelaida Fernández Sanz

9/02/2020

Introduction and Aim of the Project

This project is part of the HarvardX Data Science Proffesional Certification: Capstone. The objective of the model is to be able to predict whether a user, defining user as a visitor to a certain e-commerce website, will make a purchase on the website or not. It is about predicting the conversion per user of the e-commerce. For this, the data obtained in the realization of the article Sakar, C.O., Polat, S.O., Katircioglu, M., Neural Comput & Applic (2018) will be taken as a basis. The database consists of vectorized variables that belong to 12,330 user sessions. Each session belongs to a different user obtained during a period of one year in order to avoid any tendency to a specific campaign, specific day, user profile or specific period.

Method & Analysis

We will apply two different algoritms: SVM (Linear) and Random Forest since they are very efficient in predictive tasks that require regression and classification techniques.

Firstly, the database has been inspected to ensure that there is no missing value and to identify and inspect the variables of the database. According to the source cited above, the database consists of:

- 10 numerical variables and 8 categorical variables.
- The dependent variable, the one that we want to predict in the model is the variable "Revenue":the effectiveness in the purchase. It is a dichotomous variable: TRUE (If you buy), FALSE (do not buy).

The database is not balanced. The TRUE class is a minority (1908) compared to the FALSE class (10,422). So we will measure the performance of the model with the original database and with a balanced database to compare results and select the model with the best fit.We will use the downsampling technique in order to balance the data set. Machine learning classifiers like SVM or Random Forest do not deal very well with unbalanced training datasets as they are sensitive to the proportions of the different classes. As a consequence, these algorithms tend to favor the class with the highest proportion of observations (known as the majority class), which can lead to biased accuracy metrics.

For the SVM approach, to achieve the best possible fit, under both models (unbalanced & balanced), we will look after the best C parameters:. The C Parameter (margin), represents the complexity constant. Specifies whether the model should be more generalized or more specific. The higher the value of the parameter, the greater the specificity, but this can lead to an overfitting. We will therefore test with 10 values in this range C = [0.01 : 0.2]. For the Random Forest approach we will look after the best mtry parameter, it defines the number of variables randomly sampled as candidates at each split.

Analysis Steps

0. Download packages & Data
1. Data set Exploration
2. Data set partition
3. SVM models: 3.1 Unbalanced Data 3.1.Balanced Data
4. Random Fores 4.1.Unbalanced Data 4.1.Balanced Data

Results

0. Download packages & Data needed

```
##0.Dataset and Packages downloading
    ##Packages Download
    if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
    if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
    if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")

    library(tidyverse)
    library(caret)
    library(lattice)
    library(ggplot2)
    library(data.table)
    library(dplyr)
    library("readr")

    # Online Shoppers Purchasing Intention Dataset, csv downloaded via:
    #http://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset
    #http://archive.ics.uci.edu/ml/machine-learning-databases/00468/

    #Creating data-set:
    data<-read.csv("online_shoppers_intention (1).csv")
```

1. Dataset Exploration

Here we can see: 1. The data structure:

```
## 'data.frame':    12330 obs. of  18 variables:
##  $ Administrative         : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ Administrative_Duration: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Informational          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Informational_Duration : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ ProductRelated         : int  1 2 1 2 10 19 1 0 2 3 ...
##  $ ProductRelated_Duration: num  0 64 0 2.67 627.5 ...
##  $ BounceRates            : num  0.2 0 0.2 0.05 0.02 ...
##  $ ExitRates              : num  0.2 0.1 0.2 0.14 0.05 ...
##  $ PageValues             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ SpecialDay             : num  0 0 0 0 0 0 0.4 0 0.8 0.4 ...
##  $ Month                  : chr  "Feb" "Feb" "Feb" "Feb" ...
##  $ OperatingSystems       : int  1 2 4 3 3 2 2 1 2 2 ...
```

```
##  $ Browser                : int  1 2 1 2 3 2 4 2 2 4 ...
##  $ Region                 : int  1 1 9 2 1 1 3 1 2 1 ...
##  $ TrafficType            : int  1 2 3 4 4 3 3 5 3 2 ...
##  $ VisitorType            : chr  "Returning_Visitor" "Returning_Visitor" "Returning_Visitor" "Returni
##  $ Weekend                : logi  FALSE FALSE FALSE FALSE TRUE FALSE ...
##  $ Revenue                : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
```

2. The most relevant statistics:

```
##  Administrative   Administrative_Duration Informational
##  Min.   : 0.000   Min.   :   0.00         Min.   : 0.0000
##  1st Qu.: 0.000   1st Qu.:   0.00         1st Qu.: 0.0000
##  Median : 1.000   Median :   7.50         Median : 0.0000
##  Mean   : 2.315   Mean   :  80.82         Mean   : 0.5036
##  3rd Qu.: 4.000   3rd Qu.:  93.26         3rd Qu.: 0.0000
##  Max.   :27.000   Max.   :3398.75         Max.   :24.0000
##  Informational_Duration ProductRelated   ProductRelated_Duration
##  Min.   :   0.00        Min.   :  0.00   Min.   :    0.0
##  1st Qu.:   0.00        1st Qu.:  7.00   1st Qu.:  184.1
##  Median :   0.00        Median : 18.00   Median :  598.9
##  Mean   :  34.47        Mean   : 31.73   Mean   : 1194.8
##  3rd Qu.:   0.00        3rd Qu.: 38.00   3rd Qu.: 1464.2
##  Max.   :2549.38        Max.   :705.00   Max.   :63973.5
##   BounceRates         ExitRates         PageValues        SpecialDay
##  Min.   :0.000000   Min.   :0.00000   Min.   :  0.000   Min.   :0.00000
##  1st Qu.:0.000000   1st Qu.:0.01429   1st Qu.:  0.000   1st Qu.:0.00000
##  Median :0.003112   Median :0.02516   Median :  0.000   Median :0.00000
##  Mean   :0.022191   Mean   :0.04307   Mean   :  5.889   Mean   :0.06143
##  3rd Qu.:0.016813   3rd Qu.:0.05000   3rd Qu.:  0.000   3rd Qu.:0.00000
##  Max.   :0.200000   Max.   :0.20000   Max.   :361.764   Max.   :1.00000
##     Month           OperatingSystems    Browser           Region
##  Length:12330       Min.   :1.000    Min.   : 1.000   Min.   :1.000
##  Class :character   1st Qu.:2.000    1st Qu.: 2.000   1st Qu.:1.000
##  Mode  :character   Median :2.000    Median : 2.000   Median :3.000
##                     Mean   :2.124    Mean   : 2.357   Mean   :3.147
##                     3rd Qu.:3.000    3rd Qu.: 2.000   3rd Qu.:4.000
##                     Max.   :8.000    Max.   :13.000   Max.   :9.000
##   TrafficType    VisitorType         Weekend         Revenue
##  Min.   : 1.00   Length:12330       Mode :logical   Mode :logical
##  1st Qu.: 2.00   Class :character   FALSE:9462      FALSE:10422
##  Median : 2.00   Mode  :character   TRUE :2868      TRUE :1908
##  Mean   : 4.07
##  3rd Qu.: 4.00
##  Max.   :20.00
```

3. The number of completed purchases (TRUE) and the number of not completed purchased (FALSE)

```
##
## FALSE  TRUE
## 10422  1908
```

As seen in the basic explorarion of the dataset, there is no missing values and modification of the Revenue feature class is needed:

```r
#From logical to chr : Revenue
data <- data %>% mutate(Revenue = replace(Revenue, Revenue == "FALSE","KO"))
data <- data %>% mutate(Revenue = replace(Revenue, Revenue == "TRUE","GOOD"))
class(data$Revenue)
```

```
## [1] "character"
```

2. Dataset partition: Trainig & Test + Blanced Dataset

First we partition the data set: 85% Training, 15% test. Secondly we prapared the balanced data set with the downsampling method.

```r
##2.1Creating training-test dataset
  n=nrow(data)
  ind=1:n
  itraining=sample(ind,floor(n*0.85))
  itest=sample(setdiff(ind,itraining),floor(n*0.15))
  training = data[itraining,]
  testing = data[itest,]
  dim(training)
```

```
## [1] 10480    18
```

```r
dim(testing)
```

```
## [1] 1849   18
```

```r
##2.2 Creating balanced dataset: Downsampling method
    downSampled_training = downSample(x=training[, -ncol(training)],
                          y=as.factor(training$Revenue))
    downSampled_testing = downSample(x=testing[, -ncol(testing)],
                                  y=as.factor(testing$Revenue))
```

3. SVM Model

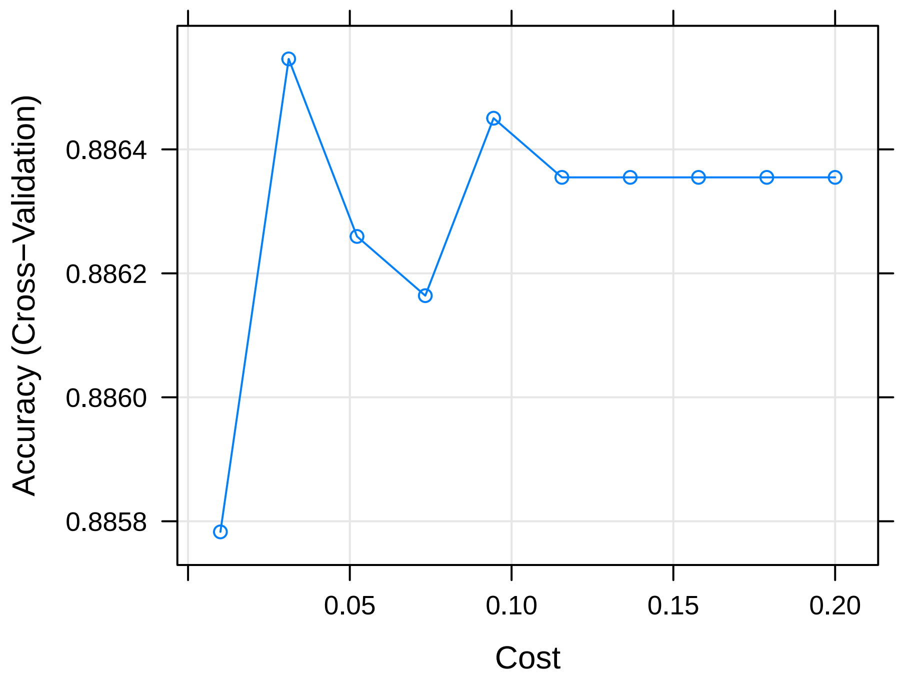
Let's start applying our SVM model. Firstly with unbalanced data, secondly with balanced data. The predictors features included in our model are the folowing:

```r
predictors = names(training)[names(training) != "Revenue"]
predictors
```

```
##  [1] "Administrative"          "Administrative_Duration"
##  [3] "Informational"           "Informational_Duration"
##  [5] "ProductRelated"          "ProductRelated_Duration"
##  [7] "BounceRates"             "ExitRates"
##  [9] "PageValues"              "SpecialDay"
## [11] "Month"                   "OperatingSystems"
## [13] "Browser"                 "Region"
## [15] "TrafficType"             "VisitorType"
## [17] "Weekend"
```

```r
##3.1. Unbalanced Data: Training svm unbalanced data. Fiting the model-> C parameter
    train_control<-trainControl(method="cv", number = 10, p = .9)
    svm_fit_u_l <- train(Revenue ~ ., method = "svmLinear", data = training,
                     trControl = train_control,
                     tuneGrid = expand.grid(C = seq(0.01, 0.2, length = 10)))
```

3.1 SVM Linear - Unbalanced Model In this plot we can see how well each C parameter performs:




```r
 # The best tuning parameter C that maximizes model accuracy
    c<-svm_fit_u_l$bestTune
    results_acc_by_c_u_l<-as_tibble(svm_fit_u_l$results[which.max(svm_fit_u_l$results[,2]),])
    results_acc_by_c_u_l
```

```
## # A tibble: 1 x 5
##        C Accuracy Kappa AccuracySD KappaSD
##    <dbl>    <dbl> <dbl>      <dbl>   <dbl>
## 1 0.0311    0.887 0.458    0.00521  0.0278
```

```r
    # Applying best C
    svm_def_u_l <- train(Revenue ~ ., method = "svmLinear", data = training,
                     trControl = train_control,
                     cost= c)
```

```
## Overall Accuracy=0.88 , Sensitivity = 0.368,  Specificity = 0.978,F = 0.859, PPV = 0.766
## , NPV = 0.888
```

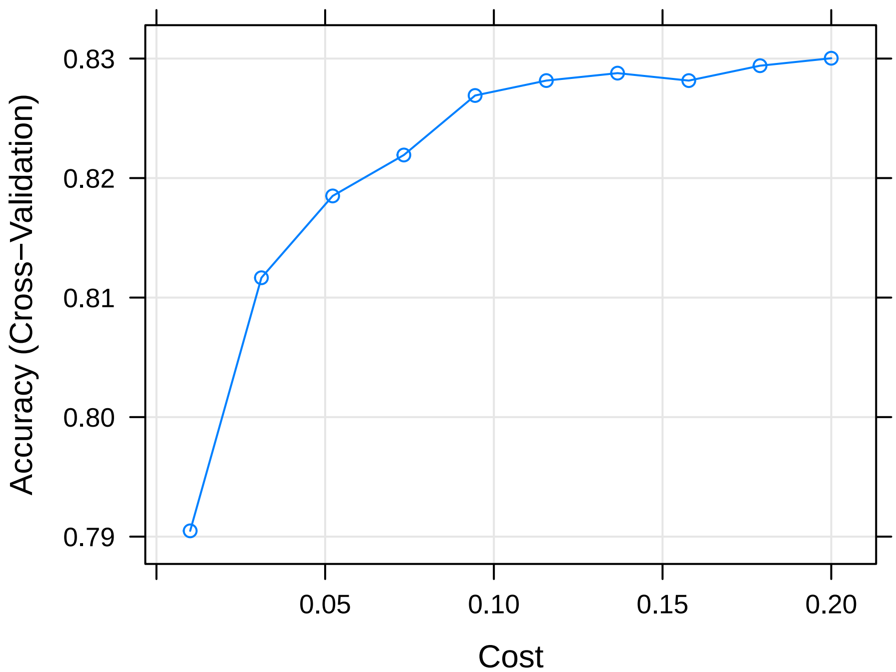
```
##           Reference
## Prediction GOOD   KO
##       GOOD  111   34
##       KO    191 1513
```

As we can see the accuracy of the balanced SVM linear model is high, but the Sensitivity is very low. The specifity, of course, is very high taking into account that we are dealing with unbalanced data. As we can see the PPV is,of course lower than the NPV.

```
##3.2. Balanced Data:SVM

  ##Training svm balanced data. Fiting the model-> C parameter
  svm_fit_b_l <- train(Class ~ ., method = "svmLinear", data = downSampled_training,
                       trControl = train_control,
                       tuneGrid = expand.grid(C = seq(0.01, 0.2, length = 10)))
```

3.2 SVM Linear - Balanced Model In this plot we can see how well each C parameter performs:




```
# The best tuning parameter C that maximizes model accuracy
    cbl<-svm_fit_b_l$bestTune
    results_acc_by_c_b_l<-as_tibble(svm_fit_b_l$results[which.max(svm_fit_b_l$results[,2]),])
    results_acc_by_c_b_l
```

```
## # A tibble: 1 x 5
##       C Accuracy Kappa AccuracySD KappaSD
##   <dbl>    <dbl> <dbl>      <dbl>   <dbl>
## 1   0.2    0.830 0.660     0.0188  0.0377
```

```r
# Aplying best C
    svm_def_b_l <- train(Class ~ ., method = "svmLinear", data = downSampled_training,
                         trControl = train_control, cost=cbl)
```

```
## Overall Accuracy=0.82 , Sensitivity = 0.752,  Specificity = 0.887,F = 0.878, PPV = 0.870
## , NPV = 0.781

##             Reference
## Prediction GOOD  KO
##       GOOD  227  34
##       KO     75 268
```

Comparing to the unbalanced model, we can see how the overall accuracy has decrease, but the sensitivity has been balanced. In this case, the PPV is greater than the NPV.

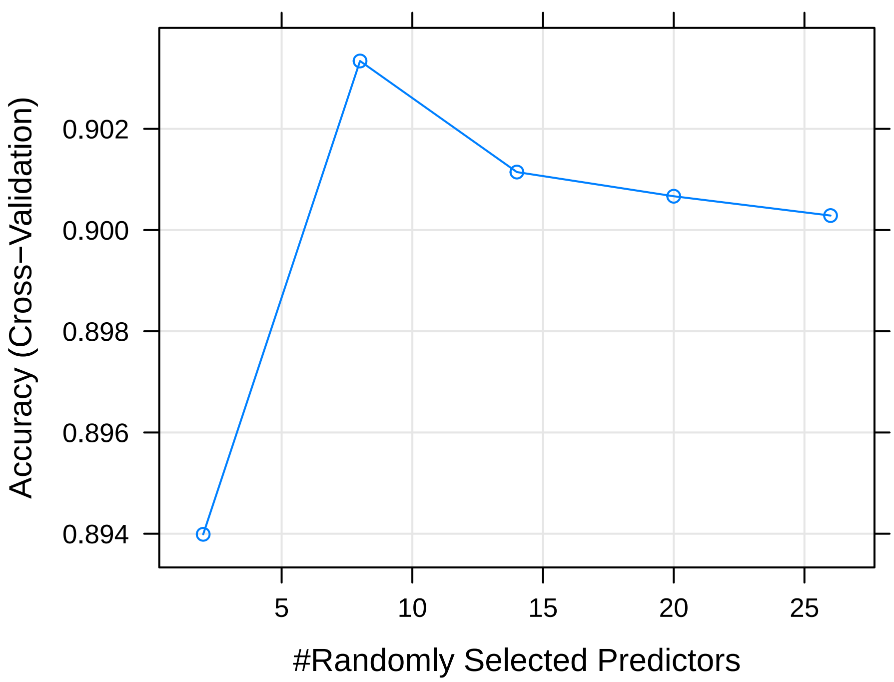
4. RF Model

Now let's take a look into the RF results.

```r

    ##3.2 Random Forest
    ##3.2.1. Training RF unbalanced data.
    train_control<-trainControl(method="cv", number = 10, p = .9)
    svm_fit_u_r <- train(Revenue ~ ., method = "rf", data = training, tuneLength=5,
                         trControl = train_control)
```

4.1 Random Forest - Unbalanced Model In this plot we can see how well each mtry parameter



performs:

```r
# The best tuning mtry that maximizes model accuracy
 mtry_u<-svm_fit_u_r$bestTune
 results_acc_mtry_u_r<-as_tibble(svm_fit_u_r$results[which.max(svm_fit_u_r$results[,2]),])
 results_acc_mtry_u_r
```

```
## # A tibble: 1 x 5
##    mtry Accuracy Kappa AccuracySD KappaSD
##   <dbl>    <dbl> <dbl>      <dbl>   <dbl>
## 1     8    0.903 0.595    0.00855  0.0398
```

```r
 # Applying best mtry
 svm_def_u_r <- train(Revenue ~ ., method = "rf", data = training,   minNode=mtry_u$mtry,trControl
```

```
## Overall Accuracy=0.90 , Sensitivity = 0.586,  Specificity = 0.966,F = 0.857, PPV = 0.770
## , NPV = 0.923
```

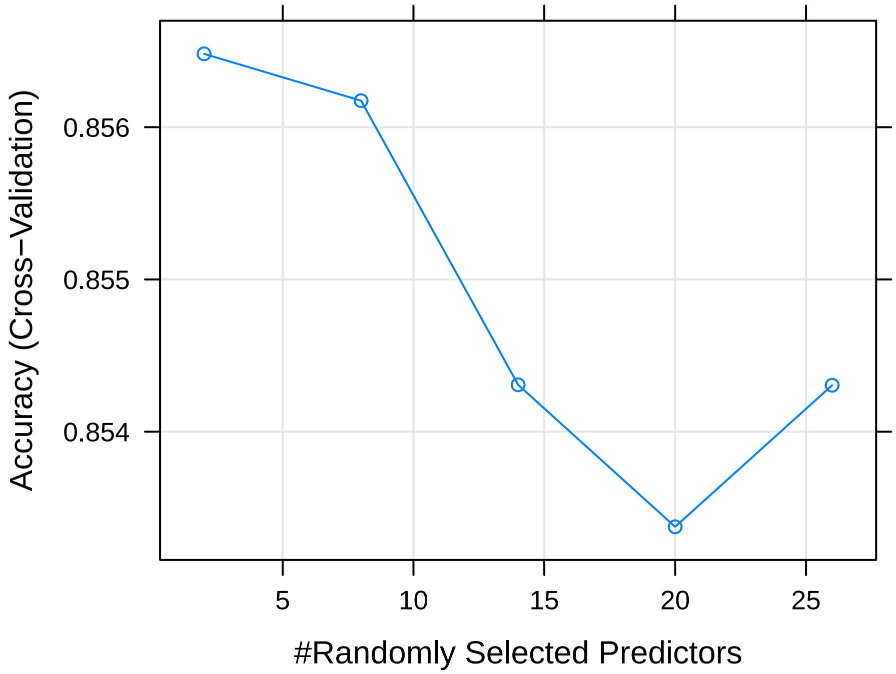
```
##           Reference
## Prediction GOOD   KO
##       GOOD  177   53
##       KO    125 1494
```

As we can see we have increase the Overal Accuracy comparing to the SVM unbalanced. Regarding Sensitivity , the unbalanced impact is not that significant compared to the SVM model. We are dealing again with unbalanced data so, as in the SVM model, the PPV is,of course lower than the NPV.

```r
##3.2. Training RF balanced data.

 ##Training svm balanced data. Fiting the model-> C parameter
 svm_fit_b_r <- train(Class ~ ., method = "rf", data = downSampled_training,
                      tuneLength=5,trControl = train_control)
```

4.2 Random Forest - Balanced Model In this plot we can see how well each mtry parameter performs:




```
# The best tuning mtry that maximizes model accuracy
cbr<-svm_fit_b_r$bestTune
results_acc_by_c_rb<-as_tibble(svm_fit_b_r$results[which.max(svm_fit_b_r$results[,2]),])
results_acc_by_c_rb
```

```
## # A tibble: 1 x 5
##    mtry Accuracy Kappa AccuracySD KappaSD
##   <dbl>    <dbl> <dbl>      <dbl>   <dbl>
## 1     2    0.856 0.713     0.0202  0.0405
```

```
# Aplying best mtry
svm_def_b_r <- train(Class ~ ., method = "rf", data = downSampled_training,
                     minNode=cbr$mtry,
                     trControl = train_control,)
```

```
## Overall Accuracy=0.85 , Sensitivity = 0.864,  Specificity = 0.831,F = 0.834, PPV = 0.837
## , NPV = 0.860
```

```
##             Reference
## Prediction GOOD  KO
##       GOOD  261  51
##       KO     41 251
```

The last model, presents higher level of Accuracy than the SVM balanced data and good balanced values of NPV and PPV.

Conclusion & Discusion

The following table presents all the relevant parameters of the four applied models

Model	Accuracy	Sensitivity	Specificity	TNrate	TPrate	F_measure
SVM-Unbalanced	0.8783126	0.3675497	0.9780220	0.8879108	0.7655172	0.8588194
SVM-Balanced	0.8195364	0.7516556	0.8874172	0.7813411	0.8697318	0.8784855
RF-Unbalanced	0.9037317	0.5860927	0.9657401	0.9227918	0.7695652	0.8565640
RF-Balanced	0.8476821	0.8642384	0.8311258	0.8595890	0.8365385	0.8338234

As we can see, the model with the highest accuracy is the RF with the original database (unbalanced) with a 90.5% accuracy. On the other hand, since it is an unbalanced database, it presents low sensitivity ratio and a low PPV ratio. The RF model with balanced bbdd has an acceptable accuracy (82%) and with very similar PPV and NPV ratios. Thus we can conclude that: RF in general has a better performance than SVM in both balanced and unbalanced databases.

Thanks for reviewing my work.

Adelaida Fernández