PAP II - Modelos de predición en empresas y gobierno mediante aprendizaje estadístico.

Juan Francisco Muñoz Elguezábal
Alumno Ingeniería Financiera
IF149833@iteso.mx

Este proyecto integrador es desarrollado como documento del Proyecto de Aplicación Profesional, actividad donde se incluye la realización del servicio social y la opción de titulación para programas de licenciatura en la universidad ITESO (Instituto Tecnológico y de Estudios Superiores de Occidente). Es realizado por un alumno de la licenciatura Ingeniería Financiera.

Índice general

Resumen

El sustento teórico para este Proyecto de Aplicación Profesional es en base a programación estadística y el área del conocimiento conocida como *Machine Learning* o *Aprendizaje Computacional*. Esto consiste en utilizar la estadística y programación para construir modelos matemáticos, algunos de forma cerrada (clasificados como analíticos) y otros de forma abierta (clasificados como computacionales o de métodos iterativos numéricos). En este proyecto se abordaron particularmente 4 que son: Support Vector Machines, Arboles de decisión y variantes, Random Forest y Procesamiento de Lenguaje Natural. Como apoyo se utilizó el sistema de control de versiones *GIT*, y como plataforma de almacenamiento en la nube se recurrió a *GITHUB*, un sitio gratuito y ampliamente conocido por su interfáz intuitiva e integración con *Markdown*, un lenguaje de escritura científica y de contenido de estructura simple. Es importante destacar que el uso del sistema operativo *Linux* y en concreto su distribución *Ubuntu 14.04 lts* ayudó a simplificar el manejo de los lenguajes de programación y el repositorio creado en *GITHUB*, así como la instalación de algunas paqueterías de *R*. También mencionar que los recursos físicos computacionales se utilizaron a un nivel mejorado, en el sentido que para las operaciones demandantes de memoria o procesamiento se utilizó una librería-*API* de conexión remota a un cluster para programación en paralelo, *H2O*.

Además de las herramientas computacionales mencionadas anteriormente, se incluye la teoría y análisis de casos para 3 metodologías citadas, *Boosted Trees* con aplicación al problema *PonPare*, *Random Forest* para resolver el problema de clasificación de variables en el problema de *Rossman*, también se presenta el caso de análisis de texto en lenguaje natural y la metodología *Word2Vec*. De igual forma y a manera de trabajo libre se presenta la estrategia de trading o inversión en el mercado *FOREX*, en esta última es donde se generó en su mayoría el conocimiento aplicado de *Machine Learning* en el área de finanzas bursátiles utilizando ingeniería financiera.

Finalmente se presentan los documentos y materiales anexos como el código genérico para la traducción del japonés al español necesario en el problema de *Ponpare*, así como la bibliografía académica y sitios de interés utilizados como referencia.

Capítulo 1

Introducción

Este trabajo representa la aplicación de los métodos de aprendizaje computacional, también conocidos como *Machine Learning*

Sección 1.1 **Objetivos**
Sección 1.2 **Justificación**
Sección 1.3 **Antecedentes**
Sección 1.4 **Contexto**

1.1. Objetivos

- **Principal:**

  Aplicar metodologías de machine learning a problemas de diversas industrias productivas y de servicios, con la finalidad de construcción de modelos para la clasificación y/o pronóstico de variables de particular importancia como las ventas, incumplimiento de pagos a crédito, etc.

- **Secundarios (de conocimiento):**

  - Utilizar metodología *Boosted trees*.
  - Utilizar metodología *Random Forest*.
  - Utilizar metodología *Natural Language Processing*.

- **Secundarios (de herramientas):**

  - Utilizar eficientemente R and Python Statistical Programming.
  - Desarrollar todo código con Git Version Control.

1.2. Justificación

La ciencia de datos en la actualidad representa una herramienta de nueva generación. Tanto para los que buscan soluciones a problemas cotidianos mediante el uso y procesamiento adecuado de datos, como para los que buscan precisamente convertirse en científicos de datos. La programación estadística es una herramienta que hace posible efectuar estudios y ser un profesionista en la ciencia de datos, para realizar programación estadística ciertamente existe un gran número de herramientas disponibles, en partícular lenguajes de programación, en este PAP y documento mostramos el uso de *R* y *Python*.

Regresar a: Capítulo 1 (Introducción)

1.3. Antecedentes

Los antecedentes en México de la ciencia de datos son difusos, a pesar de que si ha existido desarrollo en sistemas computacionales y sobre todo aplicaciones en varias industrias, ciertamente la financiera aún no goza de estos beneficios. En occidente y particularmente en Jalisco sólamente se tienen matices de esta disciplina y definitivamente ha iniciado a permearse a distintas industrias y una de ellas comienza a ser la de servicios financieros. Generalmente los modelos de predicción se realizaban con metodologías de carácter cerrado ó *de Solución Analítica*, como lo son *Box-Jenkins*, o incluso la Teoría Económica.

1.4. Contexto

Particularmente en Jalisco se ha producido el ambiente idóneo para el crecimiento y mayor uso de *Machine Learning* para dar solución a problemas complejos de clasificación de información, busqueda de patrones de comportamiento, predicción y pronóstico de variables de distinta naturaleza. Con la carrera de Ingeniería Financiera en el *ITESO*, la licenciatura en Actuaría y algunas ingenierías en sistemas y licenciaturas en matemáticas con enfóque a finanzas y programación estadística.

Regresar a: Capítulo 1 (Introducción)

Capítulo 2

Desarrollo

En este capítulo...

Sección 2.1 **Sustento teórico y metodológico**
Sección 2.2 **Planeación y seguimiento**
Sección 2.3 **Resultados**
Sección 2.4 **Reflexiones de aprendizajes**

2.1. Sustento teórico y metodológico

Se utiliza aprendizaje computacional o *Machine Learning* como metodología para procesar datos y encontrar patrones de comportamiento. Hacer uso de recursos computacionales en esta época y desde la perspectiva de ingeniería financiera es lo que hace característico a este Proyecto de Aplicación Profesional. En concreto se utilizaron técnicas de aprendizaje supervisado. Todas de métodos *tradicionales* o los que **No** son *heurísticos*.

2.2. Planeación y seguimiento

Se inició con el método de árboles de decisión para tener una metodología básica previa a las demás, a pesar de que esta es ampliamente utilizada es también considerada como la inicial y susceptible de mejoras y optimización. Posteriormente se desarrolla e implementa el método conocido como *Boosted Trees* o *árboles aumentados o estimulados*.

2.3. Resultados

Regresar a: Capítulo 2 (Desarrollo)

2.4. Reflexiones de aprendizajes

2.4.1. Implicaciones éticas

2.4.2. Aportaciones sociales

Regresar a: Capítulo 2 (Desarrollo)

Capítulo 3

Herramientas computacionales

En este capítulo...

3.1. Linux

Linux es, en términos simples, un sistema operativo. Es el software en la computadora que permite a las aplicaciones y el usuario accesar a los dispositivos en la computadora para llevar a cabo funciones o tareas deseadas. De tal manera que en estos términos linux es muy similar a otros sistemas operativos como Windows y OS X. La diferencia principal es que este sistema operativo es de código abierto, gratuito y distribuido por y para la comunidad de usuarios. Sin embargo lo anterior no ha sido impedimento para la rápida adopción de este sistema en lugares como el New York Stock Exchange y supercomputadoras privadas. Aunque sus inicios fueron mucho menos ambiciosos de lo que se podría suponer, y esque su origen se remonta a comienzo de la década de los 90-s cuando Linus Torvalds, un estudiante de ciencias computacionales, inicio con su desarrollo a manera de *hobby* el cual a finales de agosto de 1991 se convirtió en proyecto formal y publicó en un blog a la comunidad desarrolladora que buscaba colaboradores.

Para el 2008 el ecosistema Linux fue valuado en 25 billones de dólares americanos. Posiblemente debido a su portabilidad para operar bajo distintas plataformas y dispositivos electrónicos, su similaridad al sistema UNIX y que es software de licencia libre. Previo a lo cual se habían presentado eventualidades de mayor importancia para el desarrollo del sistema, un ejemplo fue lo ocurrido en el 2000, a solo 9 años después de la primera distribución de Linux, IBM decide invertir 2,000 millones de dólares en investigación y desarrollo del lenguaje, sin duda un evento significativo en la historia del sistema operativo de código abierto.

De igual manera que los sistemas operativos de Windows y OS X , Linux, y más apropiadamente algunas distribuciones de este, incluyen GUIs, esto es Graphical User Interface o Interfáz Gráfica de Usuario. Lo anterior significa que se puede tener a Linux como sistema operativo en una computadora y no tiene por qué ser complicado de utilizar ya que, según la distribución, incluye también elementos como ventanas, menus, ventanas de diálogo y elementos similares. La gran diferencia es que en Linux se tiene siempre la opción de elegir el ambiente y hacerle incluso modificaciones mayores, personalizaciones. A diferencia de windows que no es posible y en OS X es muy dificil.

En este trabajo se utilizó una computadora con sistema operativo Linux para el procesamiento de los datos, en particular la distribución de Ubuntu 14.04 lts, de manera que la mayoría de las instrucciones al sistema fueron realizadas mediante comandos de texto ingresados desde la terminal.

Algunas fuentes de información:

- Consorcio y fundación de LINUX: Linux Foundation
- Blog de Linux Foundation: Linux.com
- LINUX en Universidades de prestigio: LINUX en MIT
- LINUX en Universidades de prestigio: LINUX en Carnegie Mellon
- Soluciones de IBM en LINUX: IBM-LINUX
- Gobiernos, Organizaciones Internacionales, Empresas usando LINUX: Compare Business Products

Regresar a: Capítulo 3 (Herramientas computacionales)

3.2. Sistema de control de versiones Git

Git es un sistema de control de versiones, un conjunto de instrucciones almacenadas en un código ligero que se instala en las computadoras. Es gratuito, de distribución libre, de código abierto y ampliamente utilizado en el ambiente de desarrollo computacional, incluidas las áreas como finanzas computacionales y aprendizaje computacional. Este sistema está designado para ordenar, controlar y almacenar todas y cada una de las versiones de un proyecto computacional. Es decir, es un histórico de modificaciones realizadas a uno, varios o todos los archivos contenidos en la carpeta designada. Haciendo este control de versiones utilizando muy poca memoria, el sistema Git provee de una herramienta para controlar cambios, respaldos, y talvez la mejor de las funciones que hace posible es el código colaborativo. Mediante el uso de Git puede un grupo de desarrolladores estar trabajando en el mismo proyecto, en locaciones físicas distintas ya que es compatible con internet, controlando todos y cada uno de los cambios realizados por los integrantes del equipo.

En este proyecto se ha utilizado la plataforma GitHub

Regresar a: Capítulo 3 (Herramientas computacionales)

3.3. Programación estadística en R

Regresar a: Capítulo 3 (Herramientas computacionales)

3.4. Programación estadística en Python

Regresar a: Capítulo 3 (Herramientas computacionales)

3.5. Computo en Paralelo

Regresar a: Capítulo 3 (Herramientas computacionales)

Capítulo 4

Boosted Trees - Ponpare

En este capítulo...

Sección 4.1 **Descripción del problema**
Sección 4.2 **Exploración los datos**
Sección 4.3 **Desarrollo Teórico de modelo**
Sección 4.4 **Exploración de modelo**

4.1. Descripción del problema

Predecir cuales cupones el cliente comprará.

Regresar a: Capítulo 4 (Boosted Trees - Ponpare)

4.2. Exploración los datos

4.2.1. Descripción de archivos

Los archivos que se utilizan para este proyecto son los siguientes:

- **user_list.csv**

  Lista principal de usuarios y características de los mismos.

Column Name	Description	Type	Length	Decimal	Note
USER_ID_hash	User ID	VARCHAR2	32		
REG_DATE	Registered date	DATE			Sign up date
SEX_ID	Gender	CHAR	1		f = female m = male
AGE	Age	NUMBER	4	0	
WITHDRAW_DATE	Unregistered date	DATE			
PREF_NAME	Residential Prefecture	VARCHAR2	2		[JPN] Not registered if empty

- **coupon_list_train.csv**

  Lista principal de cupones que son considerados parte del dataset de entrenamiento

Column Name	Description	Type	Length	Decimal	Note
CAPSULE_TEXT	Capsule text	VARCHAR2	20		[JPN]
GENRE_NAME	Category name	VARCHAR2	50		[JPN]
PRICE_RATE	Discount rate	NUMBER	4	0	
CATALOG_PRICE	List price	NUMBER	10	0	
DISCOUNT_PRICE	Discount price	NUMBER	10	0	
DISPFROM	Sales release date	DATE			

- **coupon_list_test.csv**

  Lista principal de cupones que son considerados parte del dataset de prueba. La predicción realizada para la competencia debera de ser efectuada sólamente con información de éstos 310 cupones. No recibirá ningún crédito por predecir cupones comprados del dataset de entrenamiento que fueron comprados durante el periodo de prueba.

Column Name	Description	Type	Length	Decimal	Note
PURCHASE_FLG	Purchased flag	NUMBER	1	0	0:Not purchased 1:Purchased
PURCHASEID_hash	Purchase ID	VARCHAR2	128		
I_DATE	View date	DATE			Purchase date if purchased
PAGE_SERIAL		VARCHAR2			
REFERRER_hash	Referer	VARCHAR2	4000		
VIEW_COUPON_ID_hash	Browsing Coupon ID	VARCHAR2	128		
USER_ID_hash	User ID	VARCHAR2	10		
SESSION_ID_hash	Session ID	VARCHAR2	128		

- **coupon_visit_train.csv**

  El registro de visualización de los usuarios al buscar cupones durante el periodo de tiempo del dataset de entrenamiento. Esta tabla no es provista para el periodo de prueba.

Column Name	Description	Type	Length	Decimal	Note
ITEM_COUNT	Purchased item count	NUMBER	10	0	
I_DATE	Purchase date	DATE			
SMALL_AREA_NAME	Small area name	VARCHAR2	30		[JPN] User redidential area name
PURCHASEID_hash	Purchase ID	VARCHAR2	32		
USER_ID_hash	User ID	VARCHAR2	32		
COUPON_ID_hash	Coupon ID	VARCHAR2	32		

- **coupon_detail_train.csv**

  Visualización del registro de los usuarios que compraron cupones durante el periodo de entrenamiento. Esto no es provisto para el periodo de prueba.

Column Name	Description	Type	Length	Decimal	Note
SMALL_AREA_NAME	Small area name	VARCHAR2	30		[JPN]
PREF_NAME	Listed prefecture name	VARCHAR2	8		[JPN]
COUPON_ID	Coupon ID	VARCHAR2	32		

NOTA:
Debido a la naturaleza de la fuente de datos, se observó que muchos de estos están en otro idioma, japonés, por lo que ha sido necesario un procedimiento de traducción previo a iniciar con la exploración estadística. Este procedimiento se incluye en ANEXOS al final del documento.

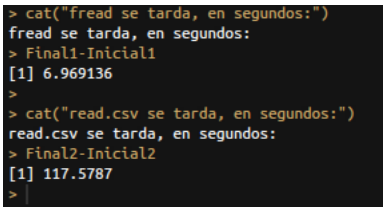
Convertir caracteres a fechas:
Para un manejo óptimo de las funcionalidades de los paquetes *lubridate,stats*, las fechas que están en formáto leible por humano *aaaa-mm-dd"* deben de estar intrínsecamente en un formáto también leible por la máquina, en este caso usaremos el formáto *POSIXct* el cual es el número de segundos transcurridos desde el momento *inicial* propuesto por éste estándar que es la fecha y hora: *1970-01-01 00:00:00*.

4.2.2. Cargar Datos

Se utiliza la función *fread*, del paquete *data.table*, similar a la función *read.table* y *read.csv* del paquete *utils*, sin embargo es más rápida y convenientemente utiliza la misma clase de datos de entrada y arroja el resultado con el mismo formáto. Se llevó a cabo una prueba, donde un archivo de 2,833,180 renglones y 8 columnas cada uno, se cargó 11 veces más rápido con la función de *fread* que con la función *read.csv*. el siguiente código muestra este proceso.

```
cat("Cargando datos para Boosted Trees - PONPARE \n")

PonPare1 <- fread("ChapBT/coupon_area_train.csv",stringsAsFactors = T)
PonPare2 <- fread("ChapBT/coupon_detail_train.csv",stringsAsFactors = T)
PonPare3 <- fread("ChapBT/coupon_list_train.csv",stringsAsFactors = T)

Inicial1 <- as.numeric(Sys.time())
PonPare4 <- fread("ChapBT/coupon_visit_train.csv",stringsAsFactors = T)
Final1 <- as.numeric(Sys.time())

Inicial2 <- as.numeric(Sys.time())
PonPare5 <- read.csv("ChapBT/coupon_visit_train.csv",stringsAsFactors = T)
Final2 <- as.numeric(Sys.time())

cat("fread se tarda, en segundos:")
Final1-Inicial1

cat("read.csv se tarda, en segundos:")
Final2-Inicial2
```

Lo que arroja como resultado lo siguiente:



Lo siguiente fue unir la información de dos data sets en uno solo. a través de la función *merge*. También se utilizó la función *fastPOSIXct* del paquete *fasttime* que aminoró de igual manera el tiempo necesario para tratar esta cantidad de datos.

```
colnames(Ponpare4)[5] <- "COUPON_ID_hash"
ChapBTTrain <- merge(Ponpare3,Ponpare4, by ="COUPON_ID_hash")
ChapBTTrain$I_DATE <- fastPOSIXct(ChapBTTrain$I_DATE)
```

Regresar a: Capítulo 4 (Boosted Trees - Ponpare)

4.3. Desarrollo Teórico de modelo

Regresar a: Capítulo 4 (Boosted Trees - Ponpare)

4.4. Exploración de modelo

- **Día de la semana en la que se realizó la visita.**

```
VISIT_WDAY  <- wday(ChapBTTrain$I_DATE)
```

- **Visitas en cupones por mes de año:**

```
CouponVisitTrain$VISIT_MONTH <- month(ChapBTTrain$I_DATE)
```

- **Visitas en cupones por género:**

```
VISIT_GENDER <- data.frame(length(which(Ponpare4$SEX_ID == "f")),
                           length(which(Ponpare4$SEX_ID == "m")))
colnames(VISIT_GENDER) <- c("f","m")
```

- **Visitas en cupones por edad:**

```
VISIT_AGE <- unique(Ponpare4$AGE[match(ChapBTTrain$USER_ID_hash,
                                       Ponpare4$USER_ID_hash)])
```

- **Año de registro de usuario:**

```
REG_YEAR <- count(year(Ponpare4$REG_DATE[match(ChapBTTrain$USER_ID_hash,
                                               Ponpare4$USER_ID_hash)]))
```

Regresar a: Capítulo 4 (Boosted Trees - Ponpare)

Capítulo 5

Random Forest - Rossman

En este capítulo...

5.1. Descripción del problema

Objetivo: Prononsticar las ventas de 1115 farmacias de la cadena Rossman localizadas por todo Alemania. Utilizando información y datos provenientes de puntos de venta, promociones y datos de competidores.

Rossmann opera más de 3.000 farmacias en 7 países europeos. Los Gerentes de las tiendas Rossmann tienen la tarea de predecir sus ventas diarias para las próximas seis semanas de operación. Naturalmente se puede pensar que las ventas en tienda son influenciadas por muchos factores, incluyendo las promociones, la competencia, la dinámica social como periodos escolares, los días festivos estatales e incluso las estaciones del año.

Regresar a: Capítulo 5 (Random Forest - Rossman)

5.2. Exploración los datos

5.2.1. Básica inicial

En los datos de *Entrenamiento* se tiene un número de tiendas de *1,115* de las cuales se recabaron datos desde el *2013-01-01* al *2015-07-31*. Así mismo al realizar la exploración se ha encontrado que el mayor número de ventas registradas en una tienda fue de *41,551*, esto en la tienda con ID: *909*. Por el contrario, la tienda que menos ventas registró fue la correspondiente al ID: *652* debido a que se registran sólo *46*.

5.2.2. Ajuste de datos

La información de fechas en las columnas *Date* y *Store* se encuentran almacenadas como un objeto tipo *Factor*, el siguiente codigo es para convertirlas a individuales, tanto para el conjunto de entrenamiento, *train*, como para el de prueba *test*. También se hace una transformación logarítmica a la cifra de ventas, con la finalidad de reducir el efecto de los datos atípicos que de manera no formal se encontraron en la exploración inicial.

```
ChapRFtrain[,Date:= as.Date(Date)]
ChapRFtest[ ,Date:= as.Date(Date)]

ChapRFtrain[,month:= as.integer(format(Date, "%m"))]
ChapRFtrain[,year:=  as.integer(format(Date, "%y"))]
ChapRFtrain[,Store:= as.factor(as.numeric(Store))]

ChapRFtest[,month:= as.integer(format(Date, "%m"))]
ChapRFtest[,year:=  as.integer(format(Date, "%y"))]
ChapRFtest[,Store:= as.factor(as.numeric(Store))]

ChapRFtrain[,logSales:=log1p(Sales)]
```

5.2.3. Uso de libreria H2O

Transcripción de documentación oficial: *R scripting functionality for H2O, the open source math engine for big data that computes parallel distributed machine learning algorithms such as generalized linear models, gradient boosting machines, random forests, and neural networks (deep learning) within various cluster environments*

Lo que significa que se utilizará para efectuar *Parallel Distributed Machine Learning Algorithms* o Algoritmos de Aprendizaje Computacional y Distribución Paralela.

5.2.4. Inicializar Cluster

```
h2o.init(nthreads=-1, max_mem_size='6G')
trainHex <- as.h2o(ChapRFtrain)
```

5.2.5. Establecer variables y entrenar modelo

```
features <- colnames(ChapRFtrain)[!(colnames(ChapRFtrain)
%in% c("Id","Date","Sales","logSales","Customers"))]

rfHex <- h2o.randomForest(x=features,y="logSales",ntrees=100,max_depth=30,
nbins_cats = 1115,training_frame=trainHex)
summary(rfHex)
```

5.2.6. Datos de prueba

```
testHex <- as.h2o(ChapRFtest)
predictions <- as.data.frame(h2o.predict(rfHex,testHex))
pred <- expm1(predictions[,1])
summary(pred)
```

Regresar a: Capítulo 5 (Random Forest - Rossman)

5.3. Desarrollo Teórico de modelo

Regresar a: Capítulo 5 (Random Forest - Rossman)

5.4. Desempeño de modelo

Regresar a: Capítulo 5 (Random Forest - Rossman)

Capítulo 6

Natural Language Processing - Word2Vec

En este capítulo...

Sección 6.1 **Descripción del problema**
Sección 6.2 **Exploración los datos**
Sección 6.3 **Desarrollo Teórico de modelo**
Sección 6.4 **Desempeño de modelo**

6.1. Descripción del problema

Regresar a: Capítulo 6 Natural Language Processing - Word2Vec

6.2. Exploración los datos

- Cuenta en Twitter.

```
username = '@iffranciscome'
```

- Registro de APP.

```
https://bitbucket.org/quant-ai/twittermining
```

- Generar llaves.

```
consumer_key    = '4TJeb1mqGw22VmxWd8w7gf3Tk'
consumer_secret = 'uvwY6vtdfxzbsbUOS14sHQMfZgKXOcRyAi7O41XbdEkZWVKXhc'
access_token    = '3288299311-sk9rkLFG0bXoeZbVbV4mJN7lmLCuUqweMhkO7BV'
access_token_secret = 'FhGj0ab2j7b7UN5LLjgeZ3ZBeTyCrkt0T6dGe6IeYAoVj'
```

Cargar Librerías

```
import tweepy as tp # API para Twitter https://github.com/tweepy/tweepy
import numpy  as np #
import pandas as pd # Manejo de datos
import json   as js # Manejo de datos tipo JSON
import nltk
import re

from nltk.tokenize import word_tokenize   # Importar funcion tokenizar.
from nltk.corpus import stopwords         # Importar lista "stop word".
from bs4 import BeautifulSoup             # Extraer datos de archivos HTML y XML.
from sklearn.feature_extraction.text import CountVectorizer
```

Autentificacion y Usuarios

```
auth = tp.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api  = tp.API(auth)

Periodicos = {'Nombre' : ['Wall Street Journal','Bloomberg Markets','CNBC'],
              'Cuenta' : ['@WSJmarkets','@markets','@CNBC']}

Profesion  = {'Nombre' : ['Guillermo Barba','Jim Cramer','Tom Keene'],
              'Cuenta' : ['@memobarba','@jimcramer','@tomkeene']}

Bolsas     = {'Nombre' : ['Dow Jones','BMV','CME Group'],
              'Cuenta' : ['@DowJones','@GrupoBMV','@CMEGroup']}

DFPer = pd.DataFrame(Periodicos)
DFPro = pd.DataFrame(Profesion)
DFBol = pd.DataFrame(Bolsas)
```

Peticiones de información

```
DFPer['Seguidores'] = 0
DFPer['Seguidores'][0] = len(api.followers_ids(DFPer['Cuenta'][0]))
DFPer['Seguidores'][1] = len(api.followers_ids(DFPer['Cuenta'][1]))
DFPer['Seguidores'][2] = len(api.followers_ids(DFPer['Cuenta'][2]))

DFPro['Seguidores'] = 0
DFPro['Seguidores'][0] = len(api.followers_ids(DFPro['Cuenta'][0]))
DFPro['Seguidores'][1] = len(api.followers_ids(DFPro['Cuenta'][1]))
DFPro['Seguidores'][2] = len(api.followers_ids(DFPro['Cuenta'][2]))

DFBol['Seguidores'] = 0
DFBol['Seguidores'][0] = len(api.followers_ids(DFBol['Cuenta'][0]))
DFBol['Seguidores'][1] = len(api.followers_ids(DFBol['Cuenta'][1]))
DFBol['Seguidores'][2] = len(api.followers_ids(DFBol['Cuenta'][2]))

guarda = api.user_timeline(DFPer['Cuenta'][0])[1].created_at

Tweets0 =  [t.text for t in api.user_timeline(DFPer['Cuenta'][0])]
Tweets1 =  [t.text for t in api.user_timeline(DFPer['Cuenta'][1])]
Tweets2 =  [t.text for t in api.user_timeline(DFPer['Cuenta'][2])]

Tweets3 =  [t.text for t in api.user_timeline(DFPro['Cuenta'][0])]
Tweets4 =  [t.text for t in api.user_timeline(DFPro['Cuenta'][1])]
Tweets5 =  [t.text for t in api.user_timeline(DFPro['Cuenta'][2])]

Tweets6 =  [t.text for t in api.user_timeline(DFBol['Cuenta'][0])]
Tweets7 =  [t.text for t in api.user_timeline(DFBol['Cuenta'][1])]
Tweets8 =  [t.text for t in api.user_timeline(DFBol['Cuenta'][2])]
```

Construcción de vocabulario

```
review_text = BeautifulSoup(Tweets0[1]).get_text()
letters_only = re.sub("[^a-zA-Z]", " ", review_text)
words = letters_only.lower().split()
stops = set(stopwords.words("english"))
meaningful_words = [w for w in words if not w in stops]
```

Regresar a: Capítulo 6 Natural Language Processing - Word2Vec

6.3. Desarrollo Teórico de modelo

Regresar a: Capítulo 6 Natural Language Processing - Word2Vec

6.4. Desempeño de modelo

Regresar a: Capítulo 6 Natural Language Processing - Word2Vec

Capítulo 7

Algorithmic Trading - Oanda

En este capítulo contiene el trabajo sobre el método de *Random Forest* aplicado a datos independientes, en este caso para encontrar la significancia de distintas señales para explicar el signo del rendimiento del precio para un activo financiero.

Sección 7.1 **Descripción del problema**
Sección 7.2 **Exploración los datos**
Sección 7.3 **Desarrollo Teórico de modelo**
Sección 7.4 **Desempeño de modelo**

7.1. Descripción del problema

Las series de tiempo financieras generalmente consisten en una serie de observaciones del precio o valor de un activo financiero realizadas en distintos puntos del tiempo. cuyas principales características cualitativas es que los intervalos de tiempo transcurrido entre una observación y otra son del mismo tamaño. Ademas que para un mismo punto de tiempo no pueden existir 2 precios. Más a detalle, el hecho de que estas series al estar ordenadas en base a una variable, el tiempo en este caso, conlleva a que se presenten generalmente autocorrelaciones seriales y efectos parecidos. Si se hiciese un modelo que aborde estos fenómenos conduciendo pruebas estadísticas formales, sería un modelo de variables endógenas. En caso contrario, si se propone utilizar series de tiempo y otras variables externas al fenómeno del precio como tal, el modelo se considera como de variables exógenas. En este trabajo se utiliza el *Machine Learning* para encontrar,a partir de una serie de variables exógenas y endógenas son útiles para realizar un pronóstico del precio y/o rendimiento.

Regresar a: Capítulo 7 Algorithmic Trading - Oanda

7.2. Exploración los datos

Correr códigos de GitHub Virtualmente

```
ROandaAPI <- "https://raw.githubusercontent.com/IFFranciscoME/ROandaAPI/master/ROandaAPI.R"
downloader::source_url(ROandaAPI,prompt=FALSE,quiet=TRUE)
PRC <- "https://raw.githubusercontent.com/IFFranciscoME/DataProcessor/master/DataProcessor.R"
downloader::source_url(PRC,prompt=FALSE,quiet=TRUE)
```

Ingresar llaves para Oanda Trading

```
consumer_key    <- '4TJeb1mqGw22VmxWd8w7gf3Tk'
consumer_secret <- 'uvwY6vtdfxzbsbUOS14sHQMfZgKXOcRyAi7041XbdEkZWVKXhc'
access_token    <- '3288299311-sk9rkLFGObXoeZbVbV4mJN7lmLCuUqweMhkO7BV'
access_token_secret <-  'FhGjOab2j7b7UN5LLjgeZ3ZBeTyCrktOT6dGe6IeYAoVj'
username <- '@iffranciscome'
```

Inicializar API de Twitter

```
setup_twitter_oauth(consumer_key,consumer_secret,access_token,access_token_secret)
```

Parámetros de trading

```
AccountID   <- 1438853    # ID de cuenta (Ver Manual ROandaAPI)
TimeAlign   <- "America%2FMexico_City" # Uso horario
Token       <- "c567fab3522f33fda6a91dbfee0522f6-cdbba372874e6e69e4694f050f890273"

Ini <- Sys.Date()-300
Fin <- Sys.Date()+1

PipValue    <- 1          # Centavo por lote por 1usd.
DayAlign    <- 14         # Hora para considerar el cierre diario
AccountType <- "practice" # Tipo de cuenta.
Granularity <- "D"      # Frecuencia de muestre de precio.
TInstrument <- "USD_MXN"  # Instrumento Financiero a utilizar.
ResagosMax  <- 29         # Resagos maximos a calcular.
InitialBalance <- 20000   # Balance Inicial.
Comision    <- 1          # En Usd por cada op.
```

Solicitar Precios.

```
ListaInst   <- data.frame(InstrumentsList(AccountType,Token,AccountID))[,c(1,3)]
PreciosHist <- HisPrices(AccountType,Granularity,DayAlign,TimeAlign,Token,TInstrument,Ini,Fin)
PrecioCl    <- data.frame(PreciosHist$TimeStamp, round(PreciosHist$Close,4))
colnames(PrecioCl) <- c("TimeStamp","PrecioCl")
PrecioAct   <- ActualPrice(AccountType,Token,TInstrument)
```

Generar Resagos.

```
ResagosCl  <- data.frame(cbind(PrecioCl[,1:2],Lag(x=PrecioCl$PrecioCl,k=1:ResagosMax)))
ResagosCl  <- ResagosCl[complete.cases(ResagosCl),]
ResagosCl$TimeStamp <- ResagosCl$TimeStamp

NC  <- .95
Reg <- c()
```

Ajustar modelo.

Regresar a: Capítulo 7 Algorithmic Trading - Oanda

7.3. Desarrollo Teórico de modelo

Regresar a: Capítulo 7 Algorithmic Trading - Oanda

7.4. Desempeño de modelo

Regresar a: Capítulo 7 Algorithmic Trading - Oanda

Capítulo 8

Anexos y Bibliografía

En este capítulo...

Sección 7.1 **Descripción del problema**
Sección 7.2 **Exploración los datos**

8.1. Anexos

8.1.1. API oficial de twitter para conectividad directa.

- Pagina Oficial Desarrolladores
- Documentacion general
- Documentacion REST APIs
- Consola oficial para pruebas

8.1.2. Traducción de Japonés a Inglés.

```
coupon_list_train = read.csv("coupon_list_train.csv", as.is=T)
trans = data.frame(jp=unique(c(coupon_list_train$GENRE_NAME, coupon_list_train$CAPSULE_TEXT,
coupon_list_train$large_area_name, coupon_list_train$ken_name,
coupon_list_train$small_area_name)), en=c("Food","Hair salon","Spa","Relaxation",
"Beauty","Nail and eye salon","Delivery service","Lesson","Gift card","Other coupon",
"Leisure","Hotel and Japanese hotel","Health and medical","Other","Hotel",
"Japanese hotel","Vacation rental","Lodge","Resort inn","Guest house",
"Japanse guest house","Public hotel","Beauty","Event","Web service","Class",
"Correspondence course","Kanto","Kansai","East Sea","Hokkaido","Kyushu-Okinawa",
"Northeast","Shikoku","Chugoku","Hokushinetsu","Saitama Prefecture","Chiba Prefecture",
"Tokyo","Kyoto","Aichi Prefecture","Kanagawa Prefecture","Fukuoka Prefecture",
"Tochigi Prefecture","Osaka prefecture","Miyagi Prefecture","Fukushima Prefecture",
"Oita Prefecture","Kochi Prefecture","Hiroshima Prefecture","Niigata Prefecture",
"Okayama Prefecture","Ehime Prefecture","Kagawa Prefecture","Tokushima Prefecture",
"Hyogo Prefecture","Gifu Prefecture","Miyazaki Prefecture","Nagasaki Prefecture",
"Ishikawa Prefecture","Yamagata Prefecture","Shizuoka Prefecture","Aomori Prefecture",
"Okinawa","Akita","Nagano Prefecture","Iwate Prefecture","Kumamoto Prefecture",
"Yamaguchi Prefecture","Saga Prefecture","Nara Prefecture","Mie","Gunma Prefecture",
"Wakayama Prefecture","Yamanashi Prefecture","Tottori Prefecture","Kagoshima prefecture",
"Fukui Prefecture","Shiga Prefecture","Toyama Prefecture","Shimane Prefecture",
"Ibaraki Prefecture","Saitama","Chiba","Shinjuku, Takadanobaba Nakano - Kichijoji",
"Kyoto","Ebisu, Meguro Shinagawa","Ginza Shinbashi, Tokyo, Ueno","Aichi",
"Kawasaki, Shonan-Hakone other","Fukuoka","Tochigi","Minami other",
"Shibuya, Aoyama, Jiyugaoka","Ikebukuro Kagurazaka-Akabane",
"Akasaka, Roppongi, Azabu","Yokohama","Miyagi","Fukushima","Much","Kochi",
"Tachikawa Machida, Hachioji other","Hiroshima","Niigata","Okayama","Ehime","Kagawa",
"Northern","Tokushima","Hyogo","Gifu","Miyazaki","Nagasaki","Ishikawa","Yamagata",
"Shizuoka","Aomori","Okinawa","Akita","Nagano","Iwate","Kumamoto","Yamaguchi","Saga",
"Nara","Triple","Gunma","Wakayama","Yamanashi","Tottori","Kagoshima","Fukui","Shiga",
"Toyama","Shimane","Ibaraki"), stringsAsFactors = F)

coupon_list_train = read.csv("coupon_list_train.csv", as.is=T)
names(trans)=c("jp","en_capsule")
coupon_list_train=merge(coupon_list_train,trans,by.x="CAPSULE_TEXT",by.y="jp",all.x=T)
names(trans)=c("jp","en_genre"); coupon_list_train=merge(coupon_list_train,
trans,by.x="GENRE_NAME",by.y="jp",all.x=T)
names(trans)=c("jp","en_small_area"); coupon_list_train=merge(coupon_list_train,
trans,by.x="small_area_name",by.y="jp",all.x=T)
names(trans)=c("jp","en_ken"); coupon_list_train=merge(coupon_list_train,trans,
by.x="ken_name",by.y="jp",all.x=T)
names(trans)=c("jp","en_large_area"); coupon_list_train=merge(coupon_list_train,trans,
by.x="large_area_name",by.y="jp",all.x=T)
write.csv(coupon_list_train, "coupon_list_train_en.csv", row.names = F)
```

```
coupon_area_train = read.csv("coupon_area_train.csv", as.is=T)
names(trans)=c("jp","en_small_area"); coupon_area_train=merge(coupon_area_train,trans,
by.x="SMALL_AREA_NAME",by.y="jp",all.x=T)
names(trans)=c("jp","en_pref"); coupon_area_train=merge(coupon_area_train,trans,
by.x="PREF_NAME",by.y="jp",all.x=T)
write.csv(coupon_area_train, "coupon_area_train_en.csv", row.names = F)
```

Regresar a: Sección 8.1 Anexos

8.2. Bibliografía

- *An Introduction to Statistical Learning with Applications in R* (Hastie,Tibshirani,Witten,James), 2013.

Regresar a: Sección 8.1 Anexos