Universidad Carlos III de Madrid

APLICACIÓN DEL APRENDIZAJE AUTOMÁTICO EN LA INDUSTRIA ENERGÉTICA

Práctica 1: Predicción de la radiación solar en plantas fotovoltaicas

AGENDA

1. **Objetivos**
2. **Problema**
3. **Datos**
4. **Desarrollo de la práctica**
   - Actividades
   - Tareas
   - Entregables
   - Planificación

Contenidos: Ricardo Aler Mur y Concepción García Diéguez

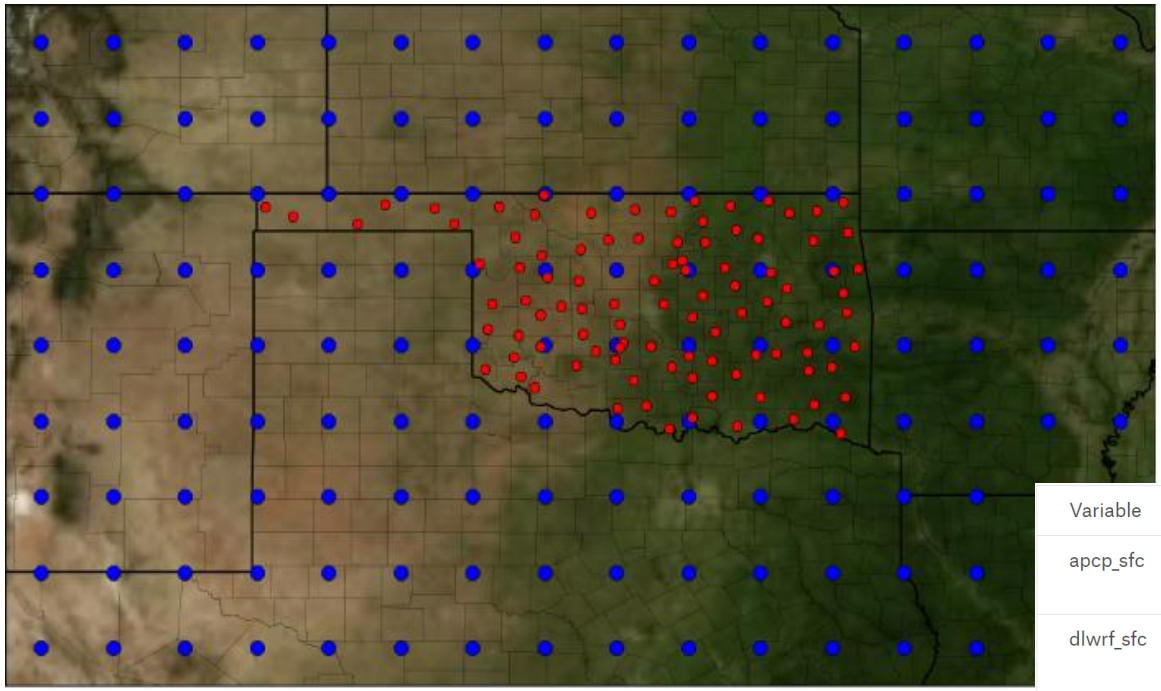
Objetivos

- Conocer un contexto de aplicación real.
- Ejercitarnos en el análisis de datos y la implementación de algoritmos de regresión en **MLR** para adquirir criterio en la aplicación de los mismos, tanto para **métodos de entrenamiento simples como avanzados**.
- Explorar las actividades de la **metodología** del AA para construir y comparar modelos.
- https://www.kaggle.com/c/ams-2014-solar-energy-prediction-contest/data (datos modificados)

Problema

- A finales del 2015 se estimó que el **23,7%** de la energía eléctrica mundial se produjo mediante fuentes de **energía renovables**
- Uno de los mayores problemas que tiene la energía solar es su **variabilidad e incertidumbre**. Las empresas productoras necesitan una estimación diaria para las siguientes 24 horas. Por ello es importante tener una predicción lo más acertada posible.
- Nuestro principal objetivo será **predecir la radiación solar** diaria en una planta solar de Oklahoma a partir de **predicciones de variables meteorológicas del día anterior**, usando **MLR**.

Problema



ROJO Estaciones fotovoltaicas

AZUL Rejilla de predicciones meteorológicas (NWP Numerical Weather Prediction)

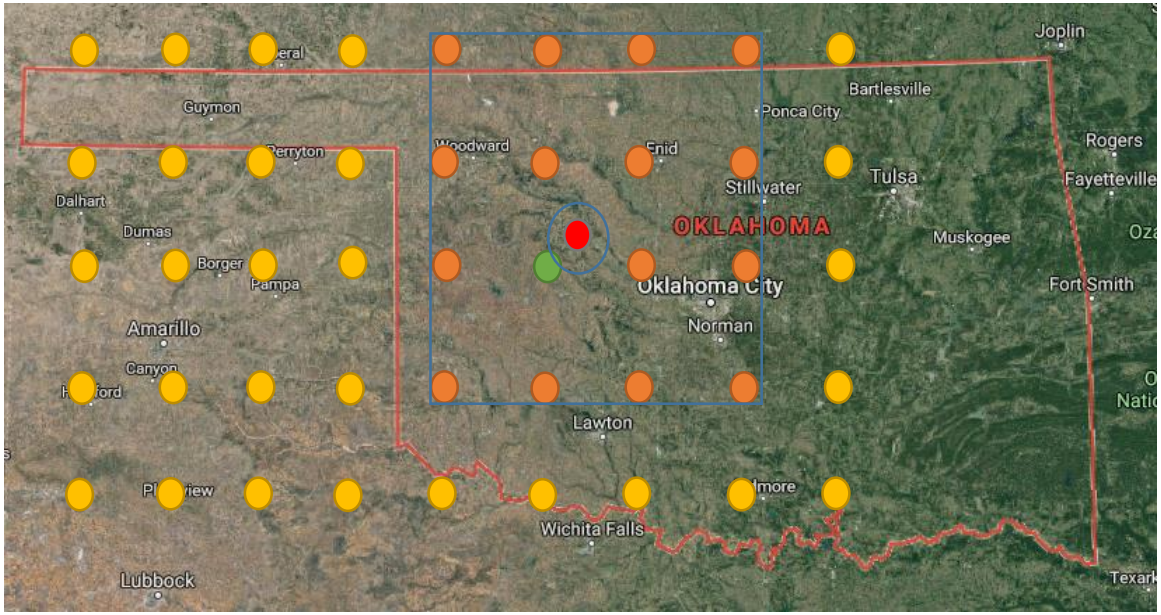
Variable	Description
apcp_sfc	3-Hour accumulated precipitation at the surface
dlwrf_sfc	Downward long-wave radiative flux average at the surface
dswrf_sfc	Downward short-wave radiative flux average at the surface
pres_msl	Air pressure at mean sea level
pwat_eatm	Precipitable Water over the entire depth of the atmosphere
spfh_2m	Specific Humidity at 2 m above ground
tcdc_eatm	Total cloud cover over the entire depth of the atmosphere
tcolc_eatm	Total column-integrated condensate

Problema consiste en encontrar modelos de aprendizaje automático *f*, de tal manera que:

- Si atributos = variables meteorológicas predichas el día anterior para el día siguiente (mediante NWP)
- Y *salida* = radiación solar acumulada día siguiente
- Ocurra que: *salida ~ f(atributos)*

Datos

Rejilla predicción. Sistema GFS (Global Forecast System)



- Rejilla completa
- Nodo más cercano
- Central solar

- La superficie terrestre está dividida horizontalmente en una malla que es de 35 o 70 kilómetros2 por celda; en la vertical la atmósfera está dividida en 64 capas y en la dimensión temporal, el GFS produce predicciones de 3 en 3 horas vista hasta las primeras 192 horas (H+3,6,9...192) y después con una resolución de 12 horas (H+204,216, 228...384).
- Variables-> temperatura media, máxima, mínima, precipitación, nubosidad, etc.
- Modelo con cobertura global cuyas predicciones están disponible gratuitamente .

Global Forecast System - Wikipedia, la enciclopedia libre

Datos

Simulando in GFS

- **Atributos de entrada**: predicciones del día siguiente de 15 variables meteorológicas. Los datos han sido generados por una simulación de ecuaciones de la atmósfera denominada NWP (Numerical Weather Prediction).
- Dichas variables han sido generadas para cinco momentos del día siguiente: 12h, 15h, 18h, 21h, 24h UTC.
- Hay 15 variables, generadas para 5 momentos del día siguiente = **75 atributos de entrada.**
- Ej: *apcp_sf1_1* significa "3-Hour accumulated precipitation at the surface". Es una variable predicha para el primero de los momentos del día siguiente. *apcp_sf3_1* sería para el tercer momento del día siguiente.
- Ej: dswrf_s2_1 significa "Downward short-wave radiative flux average at the surface".
- El significado de las variables está aquí: https://www.kaggle.com/c/ams-2014-solar-energy-prediction-contest/data.
- Importante: los datos que se van a utilizar en la práctica no son los originales, sino que tienen ligeras modificaciones.
- Es la última columna de los datos, y se denomina "**salida**". Es la radiación solar acumulada durante todo el día.

4. DESARROLLO DE LA PRÁCTICA

Actividades

- Vamos a suponer que estamos en una competición. Así, tenemos:
  - Un conjunto de datos disponibles (12 años) (disp_1.rds). Con ellos:
    - Haremos varias pruebas para obtener **el mejor modelo posible** (model selection). Esto incluye todo tipo de pruebas sobre actividades de la metodología: ¿cuál es el mejor método para preprocesar?, ¿cuál es el mejor método de construcción de modelos? ¿cuáles son los mejores hiper-parámetros para cada método? etc.
    - Obtendremos una estimación del comportamiento de lo que podría obtener el modelo en la competición.
    - Construiremos el modelo final con el que competiremos.
  - Un conjunto de datos para competir (2 años adicionales) (compet_1.rds), sin la variable de respuesta. Con ellos:
    - Aplicaremos el modelo final para obtener las predicciones que enviaríamos a la competición. Dado que realmente no hay tal competición, entregaréis las predicciones en un fichero.

Tareas

- Un conjunto de datos disponibles 1994-2003 (trainstxx.rds). Con ellos:
  - Model selection: haremos varias pruebas para obtener **el mejor modelo posible.** Esto incluye todo tipo de pruebas sobre actividades de la metodología: ¿cuál es el mejor método para preprocesar?, ¿cuál es el mejor método de construcción de modelos? ¿cuáles son los mejores hiper-parámetros para cada método? etc.
  - Obtendremos una estimación del comportamiento de lo que podría obtener el modelo en la competición.
  - Construiremos el modelo final con el que competiremos.

- Como comentamos antes, para hacer esta parte, disponemos de 9 años (desde el dato 1 hasta el 9*365).
- En general, lo que vamos a hacer en esta parte es un montón de experimentos, incluyendo ajuste de hiper-parámetros. Compararemos todos esos experimentos y nos quedaremos con el mejor. Para comparar todos los experimentos, usaremos los tres últimos años de los 9. Es decir usaremos los datos desde el 6*365+1 hasta el 9*365 para evaluar (validación) y comparar experimentos. Y desde el dato 1 hasta el 6*365 (6 años) para entrenar modelos.
- Para hacerlo, crearemos una tarea, con los datos de los 9 años.
- Usaremos como evaluación externa Holdout, pero queremos usar explícitamente los tres últimos años para evaluar y los seis primeros para entrenar. Así:

```
train_validation_partition <- rsmp("custom")
train_validation_partition$instantiate(mytask,
        train = list(1:(6*365)),
        test = list((6*365+1):mytask$nrow()))
```

- Utilizaremos como métrica relative absolute error (RAE)

El custom resampling no funciona al hacer ajuste de hiper-parámetros en mlr3 con AutoTuner$new . Usad el siguiente desc_inner (sin hacer instantiate), en lugar del custom que había que usar originalmente:

```
source("ResamplingHoldoutOrder.R")
desc_inner <- rsmp("holdoutorder",ratio=6/9)
```

(Ver Ajuste Hiper-parámetros)

Actividades

- Un conjunto de datos disponibles 1994-2003 (trainstxx.rds). Con ellos:
  - Model selection: haremos varias pruebas para obtener el mejor modelo posible. Esto incluye todo tipo de pruebas sobre actividades de la metodología: ¿cuál es el mejor método para preprocesar?, ¿cuál es el mejor método de construcción de modelos? ¿cuáles son los mejores hiper-parámetros para cada método? etc.
  - Obtendremos una estimación del comportamiento de lo que podría obtener el modelo en la competición.
  - Construiremos el modelo final con el que competiremos.

- Para obtener esta estimación reservaremos los tres últimos años. Es decir, nos olvidamos de esos datos hasta casi el final de la práctica, cuando tengamos el mejor modelo posible. Esos datos son los que van desde el dato (9*365+1) hasta el final (12*365).
- Por tanto, para hacer la parte principal de la práctica (model selection) disponemos de 9 años (desde el dato 1 hasta el 9*365).
- Utilizaremos como métrica relative absolute error (RAE)

Actividades

El objetivo principal de la práctica es obtener un modelo final que tenga el mínimo error posible.

1. Haremos EDA. Dado que la información a obtener es sencilla, prefiero que lo hagáis sin usar DataExplorer. Es importante determinar qué atributos son factores, cuales ordinales, si hay NAs, etc, para saber qué preprocesos realizar para cada método.
2. Determinación del mejor método de imputación y de escalado, usando únicamente knn.
3. Comparar varios métodos de entrenamiento:
   - reg.lm, knn, rpart, cubist, svm con kernel radial (gausiana)
   - Random Forest y Boosting
   - Todos ellos sin y con ajuste de hiper-parámetros (excepto reg.lm y cubist), para ver si el ajuste mejora el error.
4. Una vez elegido el método que funciona mejor:
   - Evaluar para estimar comportamiento futuro, usando los datos que al principio habíamos reservado para ello (los tres últimos años).
   - Construir el modelo final.
   - Usar el modelo final para calcular predicciones sobre el conjunto "compet_xx.rds"
5. Investigación adicional. Se trata de encontrar información sobre el método de ajuste de hiper-parámetros denominado “hyperband”. Describidlo en la memoria usando vuestras palabras, y usadlo para ajustar hiper-parámetros de un método, el que elijáis, describiendo las ventajas que se observan al usarlo.

Entregables

Se pide **entregar** un fichero **zip** que contenga:

- el fichero o ficheros con el código R (script o .Rmd).
- el fichero de predicciones del modelo final sobre el conjunto compet_xx.rds
- el modelo final (se puede guardar en un fichero con saveRDS).
- una memoria describiendo los resultados y lo que se pida en el enunciado (el formato puede ser cualquier de pdf, doc, html).
- Un video de no más de 5 minutos exponiendo las conclusiones de la memoria y vuestra valoración sobre la práctica

Nota importante:

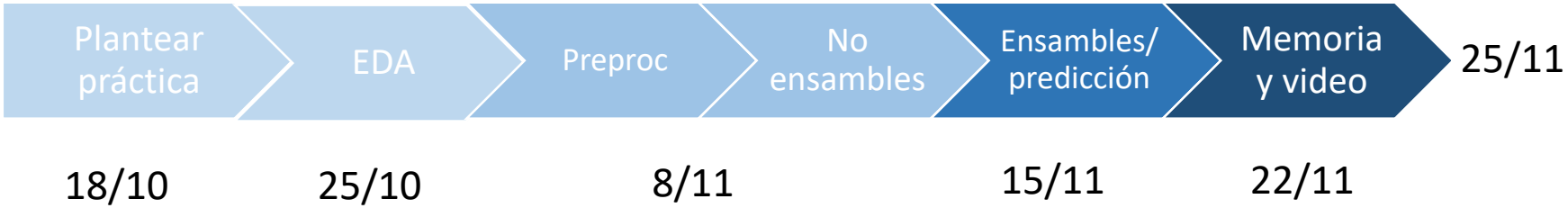
- Cada grupo debe usar la estación (punto rojo) que le corresponda (fichero adjunto práctica)
- Los resultados tienen que ser reproducibles. Como sabéis, para ello tenéis que usar *set.seed()* en los lugares apropiados. Pero en lugar de usar *set.seed(0)*, tenéis que usar set.seed con el NIA de uno de los miembros del grupo. Ejemplo, si vuestro NIA es 100345719, entonces tenéis que poner *set.seed(100345719)*.

Planificación

- Entrega: A través de Aula Global hasta el 25/11/22.
- Puntuación: sobre 4 puntos.
- Equipos de trabajo: Si se cambian los equipos de la práctica anterior comunicación por correo-e.
- Clases de prácticas:
  - 18/10.
  - 25/10.
  - 1/11: fiesta.
  - 8/11.
  - 15/11.
  - 22/11.
- Dudas fuera de clase a través de Aula global.

5. ANEXOS

PLANIFICACIÓN



EDA

(0.25 puntos) Exploratory Data Análisis: Hacer un EDA (Exploratory Data Analysis) sencillo (No DataExplorer). Se trata de saber:

- Cuántos datos (instancias) y atributos hay.
- De qué tipo son los atributos.
- Si tienen missing values (NA's) y qué proporción de sus valores son NA's.
- Si hay atributos constantes.
- Un plot de la variable de respuesta a lo largo del tiempo (y su interpretación).

```
# Lectura de datos
Entrenamiento <- readRDS(…)

# EDA
skim(Entrenamiento)

# Serie temporal de Y
serie<-
ts(Entrenamiento$salida,freq=365,start
= c(1992,1))
plot(serie, main="Serie temporal de la
radiación solar(1992-2003)",
xlab="Año", ylab="Radiación solar")
```

Preproceso

- **Sin info-leakage**
  - -> suprimir columnas constantes
  - -> suprimir columnas con NA's>80%
- **Con info-leakage**
  - -> convertir predictores, imputar NA's (columnas NA's< 80% -univariantes/multivariantes-)y escalar cuando el método ML lo requiera

PREPROCESADO

- **(0.25 puntos) Métrica: relative absolute error (RAE):** Esta va a ser la métrica que vamos a usar para evaluar los modelos. Describidla usando vuestras propias palabras y encontrad cómo usarla en mlr3.

```
# Métrica:
medida<-msr("regr.rae")
```

- **(0.25 puntos) ¿Cuál es el mejor método de imputación y de escalado?**: Se trata de comparar varios métodos de imputación y de escalado, y determinar cuál es la mejor opción. Lo haremos **sólo para el vecino más cercano** con hiper-parámetros por omisión (**default**). En este apartado, me gustaría ver dos tablas, una donde se evalúan distintas maneras de imputar, y otra donde se evalúan las distintas maneras de escalar (normalizar). Por evaluar me refiero a calcular el RAE en el conjunto de validación.

Estudio del mejor método de imputación y escalado para KNN
- Learner
  - knn_learner<-lrn("regr.kknn",…) o knn_learner<-lrn("regr.fnn",…)
- Train Task
  - res_desc <- rsmp("custom")
  - KNN_train<-list(1:(6*365)) # 6 años
  - KNN_test<-list((6*365+1):(9*365)) # 3 años
- Imputacion_categ: imputemode, imputesample, classif_learner_rpart
- Imputacion_numerica=imputemean, imputemedian, imputehist. regr_learner_rpart
- Escalado=scale“, scalerange", scalemaxabs

PREPROCESADO

Preproceso

- **Sin info-leakage** (con o sin pipeline)
  -> suprimir columnas constantes
  -> suprimir columnas con NA's>80%
- **Con info-leakage** (con pipeline)
  -> convertir predictores, imputar NA's (columnas NA's< 80% - univariantes/multivariantes-)y escalar cuando el método ML lo requiera

Pipeline:

mlr3book - 6 Pipelines (mlr3book-mlr--org-com.translate.goog)
https://mlr3pipelines.mlr-org.com/reference/index.html

- encode Factor Encoding (one-hot / dummy)
- po("colapply", applicator = as.integer, affect_columns = selector_type("ordered")) # Ordinal a entero
- removeconstants Remove Constant Features
- scale Center and Scale Numeric Features (estandardización)
- …

```
#########################################################################################
# Ejemplo 7.2: Hacemos los preprocesos sin info-leakage primero
#   y después usamos una secuencia para preprocesos con posible info-leakage
#   En este segundo ejemplo hacemos el preproceso sin info-leakge directamente con mlr3
#   y usamos secuencias para los de info-leakage
#########################################################################################

# Estos datos ya están dentro de mlr3 en forma de tarea
pima_task ← tsk("pima")

# Preprocesos sin info-leakage
# Ojo, esto es un ejemplo un poco artificial, porque pima no tiene ni variables categóricas,
#   ni columnas constantes
# encode es dummy-hot-encoding

preproc_inicial ← po("removeconstants") %>>%
                  po("encode")

pima_task ← preproc_inicial$train(pima_task)[[1]]

learner_name ← "classif.fnn"
fnn_learner ← lrn(learner_name)

# Preprocesos con info-leakage. Van en la secuencia
preproc = po("imputemean") %>>%
          po("scalerange")

# Añadimos el learner a la secuencia
graph = preproc %>>%
  po(fnn_learner)

graph$plot()

# Convertimos la secuencia en un learner
impute_scale_fnn_learner ← as_learner(graph)
```

ESTE ES EL MÉTODO PREFERIDO

PP sin infoleakage con pipeline:

- Construimos pipeline
- Train sobre la task con los datos

Construimos un learner con PP con infoleakage y el learner base

- Construimos pipeline con PP + learner
- Convertimos el pipeline a learner

```r
# Lo que sigue es lo habitual con cualquier learner
# Definimos un método de evaluación
res_desc ← rsmp("holdout", ratio=2/3)
set.seed(0)
res_desc$instantiate(pima_task)

# Entrenamos y evaluamos el modelo
set.seed(0)
fnn_resample ← resample(task=pima_task,
                        learner=impute_scale_fnn_learner,
                        resampling=res_desc)

# Calculamos el error
fnn_acc ← fnn_resample$aggregate(msr("classif.acc"))
print(fnn_acc)
```

Entrenamos como es habitual.

- Nuestra práctica requiere además comparar resultados con los distintos preprocesados

 # Bucles anidados para recorrer las combinaciones de i.categórica, inumérica y escalado

- for **esc** in c(“scale”, “scalerange”,”scalemaxabs")
  for **i_cat** in c("imputemode”, "imputesample, "classif_learner_rpart")
  for **i_num** in c("imputemean", "imputemedian", "imputehist", "regr_learner_rpart")
- **#Construcción pipeline** (dentro de los bucles anidados)

 # Elección de imputación numérica univariante o multivariante

```
if(i_num!="regr_learner_rpart")
        {knn_pre<-po(i_num, affect_columns=selector_type("numeric"),id=“Imp_num")}
else {knn_pre<-
        po("imputelearner",lrn("regr.rpart"),affect_columns=selector_type("numeric"), id="Imp_num")}
# Elección de imputación categórica univariante o multivariante
if(Ii_cat!="classif_learner_rpart")
        {knn_pre <-knn_pre%>>%po(i_cat, affect_columns=selector_type("factor"), id="Imp_cat")}
else {knn_pre<-knn_pre%>>
        %po("imputelearner",lrn("classif.rpart"),affect_columns=selector_type("factor") ,id=" Imp_cat")}
#Añadimos escalado y el learner a la secuencia
kn_grph <- knn_pre%>>%po(esc)%>>%po(knn_learner)
# Convertimos la secuencia en un learner Imput_
knn_learner<- as_learner(knn_graph)
```

- Entrenamos y evaluamos cada superlearner y montamos tabla de resultados para elegir triada menor error

COMPARACIÓN NO ENSEMBLES SIN AJUSTE HP

(0.25 Puntos) Evaluar varios métodos SIN ajuste de hiper-parámetros (o sea, con hiper-parámetros por omisión / default): Este apartado se hará con los siguientes métodos: **regr.*lm*, *rpart*, vecino más cercano, *cubist*, y *SVM* lineal y radial (kernel gausiano). Tabla** para mostrar los resultados. ¿criterios de comparación?: **error, tiempo de procesamiento** tiempo, ….

Regresión lineal. El learner no soporta factores ordenados ni NA's, elegir una opción de imputación o estudiar el mejor método de imputación categórica y numérica que minimiza el RAE

- lm_learner<-lrn("regr.lm")

Árboles de decisión (Rpart)

- rpart_learner<-lrn("regr.rpart")

KNN vecino más cercano, podemos utilizar los mejores valore de imptación y escalado del apartado de evaluación de imputación

- knn_learner<-lrn("regr.kknn",scale=FALSE)

Cubist. El learner no acepta NA´s, elegir una opción de imputación o estudiar el mejor método de imputación categórica y numérica que minimiza el RAE

- cubist_learner<-lrn("regr.cubist")

Maquinas de vector soporte: linear y radial El learner no soporta factores ordenados, ni factores (convertir tipos), ni NA's (elegir una opción de imputación o estudiar el mejor método de imputación categórica y numérica que minimiza el RAE)

- kernels<-c("linear","radial") #Podemos hacerlo en un bucle
- svm_learner<-lrn("regr.svm",kernel=kernels[i])

#**Evaluación**: Como nos piden evaluar con métrica RAE y entrenar con los hiperparámetros defecto la evaluación se hace con esa métrica y partición de datos "custom" con 6 años para train y 3 para validación

COMPARACIÓN NO ENSEMBLES CON AJUSTE HP

(1) Ahora vamos a hacer lo mismo, pero con ajuste de hiper-parámetros:

- Para el **vecino más cercano**, ajustad sólo el número de vecinos con ***Grid Search***.
- Para ***rpart*** **y SVM** usad ***Random Search.***

```
# Datos Outer Validation
• Outer_train<-list(1:(9*365)) # 6 años
• Outer_test<-list((9*365+1):nrow(Entrenamiento)) # 3 años
• Outer_res_desc$instantiate(Adjust_Train_task, Outer_train, Outer_test)
# Datos Inner Validation Ajuste de Hiperparámetros para AutoTurner
•  Inner_res_desc<-rsmp("holdoutorder",ratio=6/9)
```

```
# Learner y Preproceso
• ….
#Definición del espacio de búsqueda KNN Grid Search, valorar la selección de los HP opción
• kknn_space<-ps(regr.kknn.k=p_fct(levels = seq(1,50,2)))
#Learner con ajuste de hiperparámetros
• kknn_ajuste<-AutoTuner$new( learner = knn_learner, resampling = Inner_res_desc,
  measure = medida, search_space = kknn_space, terminator = trm("none"),
  tuner=tnr("grid_search"), store_tuning_instance = TRUE)
#Evaluación del learner con hiperparametros ajustados
• kknn_ajuste_resample<-
resample(Adjust_Train_task,kknn_ajuste,Outer_res_desc,store_models = TRUE)
#Grafico rae en función del valor del hiperparámetro
• autoplot(…)
```

COMPARACIÓN NO ENSEMBLES CON AJUSTE HP

(1) Ahora vamos a hacer lo mismo, pero con ajuste de hiper-parámetros:

- Para el **vecino más cercano**, ajustad sólo el número de vecinos con ***Grid Search***.
- Para ***rpart*** y **SVM** usad ***Random Search.***

```
# Datos Outer Validation
• Outer_train<-list(1:(9*365)) # 6 años
• Outer_test<-list((9*365+1):nrow(Entrenamiento)) # 3 años
• Outer_res_desc$instantiate(Adjust_Train_task, Outer_train, Outer_test)
# Datos Inner Validation Ajuste de Hiperparámetros para AutoTurner
• Inner_res_desc<-rsmp("holdoutorder",ratio=6/9)
```

```
# Learner y Preproceso...
# Definición del espacio de búsqueda rpart Ramdon Search, valorar la selección de los HP opción
• rpart_space <- ps( minsplit = p_int(lower=10, upper=60), maxdepth = p_int(lower = 20,upper = 30) )
#Learner con ajuste de hiperparámetros
• rpart_ajuste <- AutoTuner$new( learner = rpart_learner, resampling = Inner_res_desc,
  measure = medida,  search_space = rpart_space, terminator = trm("evals", n_evals = 50 ),
  tuner = tnr(“random_search"))
#Evaluación del learner con hiperparametros ajustados
• rpart_ajuste_resample <- resample(Adjust_Train_task, rpart_ajuste, Outer_res_desc,
store_models = TRUE)
#Grafico rae en función del valor del hiperparámetro
• autoplot(…)
```

COMPARACIÓN NO ENSEMBLES CON AJUSTE HP

(1) Ahora vamos a hacer lo mismo, pero con ajuste de hiper-parámetros:

- Para el **vecino más cercano**, ajustad sólo el número de vecinos con ***Grid Search***.
- Para ***rpart*** y **SVM** usad ***Random Search.***

```
# Datos Outer Validation
• Outer_train<-list(1:(9*365)) # 6 años
• Outer_test<-list((9*365+1):nrow(Entrenamiento)) # 3 años
• Outer_res_desc$instantiate(Adjust_Train_task, Outer_train, Outer_test)
# Datos Inner Validation Ajuste de Hiperparámetros para AutoTurner
•  Inner_res_desc<-rsmp("holdoutorder",ratio=6/9)
```

```
# Learner y Preproceso ...
# Definición del espacio de búsqueda SVM (radial o gausiana) Ramdon Search, valorar la selección de los HP opción
• if(kernels[i]=="radial"){svm_space <- ps( regr.svm.cost = p_dbl(lower=-3, upper=3, trafo=function(x) 10^x), regr.svm.gamma = p_dbl(lower=-3, upper=3, trafo=function(x) 10^x))}else{ svm_space <- ps(regr.svm.cost = p_dbl(lower=-2, upper=2, trafo=function(x) 10^x))}
• # Def. de terminación if(i==1){terminator <- trm("evals", n_evals = 5)}else{terminator <- trm("evals", n_evals = 20)}
#Learner con ajuste de hiperparámetros …  (pág. 24)
#Evaluación del learner con hiperparametros ajustados … (pág. 24)
#Grafico rae en función del valor del hiperparámetro  … (pág. 24)
```

KERNEL

Kernel:

$$\left(\sum_i (\alpha_i y_i) K(\vec{x}_i, \vec{x}) \right) + b = 0$$

Existen diferentes kernels, dependiendo de como se quiera medir la similitud (***K(x$_i$, x))***.

- Kernel **lineal**

$$K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^t \mathbf{y}$$

- Kernel **polinómico**

$$K(\mathbf{x}, \mathbf{y}) = (1 + \gamma \mathbf{x}^t \mathbf{y})^d$$

- Kernel **radial, gausiano** (el más común)

$$K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)$$

regr.svm

Parameters

Id	Type	Default	Levels	Range
cachesize	numeric	40		$(-\infty, \infty)$
coef0	numeric	0		$(-\infty, \infty)$
cost	numeric	1		$[0, \infty)$
cross	integer	0		$[0, \infty)$
degree	integer	3		$[1, \infty)$
epsilon	numeric	0.1		$[0, \infty)$
fitted	logical	TRUE	TRUE, FALSE	-
gamma	numeric	-		$[0, \infty)$
kernel	character	radial	linear, polynomial, radial, sigmoid	-
nu	numeric	0.5		$(-\infty, \infty)$
scale	untyped	TRUE		-
shrinking	logical	TRUE	TRUE, FALSE	-
tolerance	numeric	0.001		$[0, \infty)$
type	character	eps-regression	eps-regression, nu-regression	-

R: Support Vector Machine (r-project.org)

El parámetro gamma, el coste, la complejidad de un SVM - Análisis y Decisión (analisisydecision.es)

COMPARACIÓN ENSEMBLES: RANDOM FOREST

(0.5) Ahora probad con métodos de ensembles SIN Y CON ajuste HP

- Random Forest: con *ranger*. Ajustad sólo sus dos hiper-parámetros más importantes.
- Gradient Boosting: *catboost* o *xgboost con HP más importantes*

```
# Datos Outer Validation
• Outer_train<-list(1:(9*365)) # 6 años
• Outer_test<-list((9*365+1):nrow(Entrenamiento)) # 3 años
• Outer_res_desc$instantiate(Adjust_Train_task, Outer_train, Outer_test)
# Datos Inner Validation Ajuste de Hiperparámetros para AutoTurner
•  Inner_res_desc<-rsmp("holdoutorder",ratio=6/9)
```

```
# Learner
• Forest_learner<-lrn("regr.ranger")
# Preproceso sin ajuste de hiperparámetros Para imputar NAs, probamos todas las posibles
combinaciones de los métodos de imputación con el fin de encontrar aquella combinación de
métodos de imputación que minimice el RAE.
# Sin ajuste HP: Entrenamiento y evaluación (similar a página 22)
# Con ajuste de HP- Definición del espacio de búsqueda. valorar la selección de los HP opción
• Forest_space <- ps( regr.ranger.num.trees = p_int(lower=300,upper=900),
  regr.ranger.mtry=p_int(lower=15,upper=25) )
• terminator <- trm("evals", n_evals = 20 )
• tuner <- tnr("random_search")
#Learner con ajuste de hiperparámetros …  (pág. 24)
#Evaluación del learner con hiperparametros ajustados … (pág. 24)
#Grafico rae en función del valor del hiperparámetro  … (pág. 24)
```

HIPERPARÁMETROS Ramdon Forest

- Hiper-parámetros: **número *M* de árboles en el ensemble, K el número de atributos del subconjunto aleatorio y n el tamaño mínimo de elementos en un nodo para seguir subdividiendo)**

Parameters

Id	Type	Default	Levels	Range
alpha	numeric	0.5		$(-\infty, \infty)$
always.split.variables	untyped	-		-
holdout	logical	FALSE	TRUE, FALSE	
importance	character	-	none, impurity, imp	
keep.inbag	logical	FALSE	TRUE, FALSE	
max.depth	integer	NULL		
min.node.size	integer	5		
min.prop	numeric	0.1		
minprop	numeric	0.1		
mtry	integer	-		
mtry.ratio	numeric	-		
num.random.splits	integer	1		
num.threads	integer	1		$[1, \infty)$

M: Número de árboles - **regr.ranger.num.trees**
K: número de atributos a ser tenidos en cuenta para elegir el de mayor ganancia a la hora de montar árboles - **regr.ranger.mtry**
n: número de elementos para seguir subdividiendo el árbol=equivalente a la profundidad – **dejamos por defecto**

Ranger Regression Learner — mlr_learners_regr.ranger • mlr3learners (mlr-org.com)

COMPARACIÓN ENSEMBLES: RANDOM FOREST

(0.5) Ahora probad con métodos de ensembles SIN Y CON ajuste HP

- Random Forest: con *ranger*. Ajustad sólo sus dos hiper-parámetros más importantes.
- Gradient Boosting: *catboost* o *xgboost con HP más importantes*

```
# Datos Outer Validation
• Outer_train<-list(1:(9*365)) # 6 años
• Outer_test<-list((9*365+1):nrow(Entrenamiento)) # 3 años
• Outer_res_desc$instantiate(Adjust_Train_task, Outer_train, Outer_test)
# Datos Inner Validation Ajuste de Hiperparámetros para AutoTurner
•  Inner_res_desc<-rsmp("holdoutorder",ratio=6/9)
```

```
# Learner
• Xgboost_learner<-lrn("regr.xgboost")
# Preproceso sin ajuste de hiperparámetros Para imputar NAs, probamos todas las posibles combinaciones de los métodos de imputación con el fin de encontrar aquella combinación de métodos de imputación que minimice el RAE.
# Sin ajuste HP: Entrenamiento y evaluación (similar a página 22)
# Con ajuste de HP- Definición del espacio de búsqueda. valorar la selección de los HP opción
• set.seed (0)
• Xgboost_space <- ps( regr.xgboost.nrounds = p_int(lower=1,upper=20),
  xregr.xgboost.eta=p_int(lower=1,upper=100, trafo=function(x) x/100) )
• terminator <- trm("evals", n_evals = 40 )
• tuner <- tnr("random_search")
#Learner con ajuste de hiperparámetros …  (pág. 24)
#Evaluación del learner con hiperparametros ajustados …  (pág. 24)
```

Ejemplos de espacios de búsqueda de hiper-parámetros

- http://mlrhyperopt.jakob-r.de/parconfigs

gbm

	id	type	lower	upper	default	trafo
maxdepth	interaction.depth	integer	1	10	1	
minsplit	n.minobsinnode	integer	5	25	10	
	n.trees	numeric	0	6.6439	5.6439	function (x) round(2^x * 10
	shrinkage	numeric	0.001	0.6	0.001	

1000 árboles

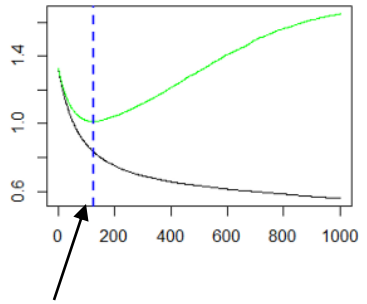
$f_t(x) = f_{t-1}(x) + v\,\alpha_t\,h_t(x)$

xgboost

	id	type	lower	upper	default	trafo
	colsample_bytree	numeric	0.3	0.7	0.5	
shrinkage	eta	numeric	0.001	0.6	0.3	
	gamma	numeric	0	10	0	
	max_depth	integer	1	10	6	
	min_child_weight	numeric	0	20	1	
	nrounds	numeric	0	8.6439	0	function (x) round(2^x * 10
	subsample	numeric	0.25	1	1	

4000 árboles

Boosting y sobreaprendizaje



- Controlar el número de árboles T en el modelo (T pequeño => menos sobreaprendizaje)
- Shrinkage (0<v<1) o learning rate: $f_t(x) = f_{t-1}(x) + v\,\alpha_t\,h_t(x)$
  - Esto consigue que el nuevo modelo tenga menos peso del que tendría y por tanto sobreajusta menos a los datos
  - v pequeños suele implicar que haya que poner más árboles (T) en el ensemble, y por tanto, el aprendizaje tarda mas
- Controlar profundidad de los árboles (modelos base). Cuanto más pequeña, menos sobreajuste. Valores a explorar: 1-10
- Stochastic gradient boosting (siguiente transparencia)