Universidad de Buenos Aires
Maestría en Data Mining
DM EyF
ciclo 2019 "the Red Pill year"

Tarea para el Hogar OCHO

En esta tarea se probarán los siguientes algoritmos y librerías (lo que está en gris suave ya se probó y no se va a volver a utilizar)

Model Generation		
Algorithm	**Library**	**Library Multi threading**
CART (Decision Tree)	rpart	NO
Random Forest	ranger	Yes
Gradient Boosting	XGBoost	Yes
	LightGBM	Yes

Todos los algoritmos que generan modelos predictivos tienen parámetros, que al ser del algoritmo y no del modelo que generan se les da el nombre de hiperparámetros.
La búsqueda de los parámetros optimos para un < algoritmo, dataset, clase> se llama optimización de hiperparámetros.

Hyperparameter Optimization	
Algorithm	**Library**
Grid Search	for loops
Bayesian Optimizacion	mlrMBO

Todo modelo debe ser evaluado en datos nuevos, que no fueron vistos al momento de entrenamiento

Model Estimation https://en.wikipedia.org/wiki/Cross-validation_(statistics)	
Methodology	**Comments**
Holdout method	randomly assign data points to two sets d_0 and d_1, usually called the training set and the test set, respectively.
Monte Carlo cross-validation (Repeated random sub-sampling validation)	creates multiple random splits of the dataset into training and validation data
k-fold cross-validation	the original sample is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k - 1$ subsamples are used as training data (esto NO fue visto en clase)
Walk Forward Optimization	https://en.wikipedia.org/wiki/Walk_forward_optimization

Todas las corridas de esta tarea son en la nube en máquinas virtuales preemtive o sea mortales, que a lo sumo viven 24 horas. El alumno deberá ir controlando sus máquinas y volviendo a lanzar el proceso en caso de ser necesario, los scripts han sido confeccionados de forma que retoman desde donde quedaron al vovler a ser corridos.
Para estas corridas es necesario tomar nota del costo por hora de la máquina virtual creada y de la cantidad total de horas que llevó la corrida, la idea es calcular al final el costo total de la corrida.

Cada corrida es un experimento, y queda en un archivo llamado hiperparámetro_xxx.txt en la carpeta work. Tal como fue visto en clase para obtener la mejor la ganancia se debe :

- Ordener en forma descenente por metrica1_actual
- La ganancia en datos nuevos queda en metrica1_futuro de la primer linea

Atención, jamás se debe entregar como valor el máximo de metrica1_futuro, eso es hacerse trampa al solitario.

En las siguientes tareas se entrenará en un conjunto de n meses, donde esa cantidad de meses donde se entrena será un hiperparámetro a optimizar por la Optimización Bayesiana. Con el único propósito de acelerar el procesamiento, se hace un sampling de los datos, de la siguiente forma :

- BAJA+2 se conservan todos
- { CONTINUA, BAJA+2} solo se consevan el 10%

1. Tareas de housekeeping

    - Se han hecho cambios a la carpeta R del dropbox, bajarla nuevamente a la PC local y luego subir al bucket de Google Cloud

    - Verificar que en el bucket de Google Cloud se tienen las carpetas, estas carpetas y sus archivos se generan con los scripts que están en /R/FeatureEngineering/
        - datasets/dias/
        - datasets/ext/
        - datasets/hist/

    - Leer en que consiste la Tarea 7 antes de seguir con las siguientes tareas.

2. Correr R/rpart/rpart_tune_MBO_meses_undersampling.r

Hasta ahora la mejor ganancia que tenemos con rpart sobre 201904_dias.txt es de alrededor $ 6.100.000 . Los parámetros optimos se encontraron entrenando en los datos de 201902_dias.txt haciendo 5-fold Montecarlo.

Pues bien, antes de abandonar el algoritmo rpart para pasar a algoritmos de ensembles, vamos a intentar sacarle todo el jugo posible. Para ello entrenando en varios meses del pasado vamos lograr obtener una ganancia en 201904_dias.txt de aproximadamente $ 7.400.000

Dados unos < ventana_meses, maxdepth, minsplit, minbucket, cp > primero vamos en generar un modelo en n meses, tomados del 201812 hacia el pasado, y medir la ganancia en 201902 , esa la vamos a llamar metrica1_actual
Luego usando esos mismos parámetros entrenamos en 201902 y n meses tomados hacia el pasado, y aplicamos el modelo a 201904, a esa ganancia la llamamos metrica1_futuro

Por ejemplo, si n=8 entrenamos en la unión de los meses de {201805, 201806, 201807, 201808, 201809, 201810, 201811, 201812} y a ese modelo lo aplicamos a 201902
Como segundo paso, con esos mismos parámetros entrenamos en {201807, 201808, 201809, 201810, 201811, 201812, 201901, 201902} y a ese modelo lo aplicamos a 201904

Es la Optimización Bayesiana que resuelve toda la optimización al mismo tiempo

```
obj.fun <- makeSingleObjectiveFunction(
        name = "prueba",
        fn   = funcion_optimizar,
        par.set = makeParamSet(
            makeIntegerParam("pmaxdepth"         ,  lower=6L    , upper=  30L),
            makeNumericParam("pminbucket"        ,  lower=0.05  , upper=   0.5),
            makeIntegerParam("pminsplit"         ,  lower=1L    , upper= 200L),
            makeNumericParam("pcp"               ,  lower=0.0   , upper=   0.001),
            makeIntegerParam("pventana"          ,  lower=1L    , upper=  12L)
        ),
        has.simple.signature = FALSE,
        global.opt.value = -1
        )
```

El alumno que considera perfectibles los lower/upper elegidos para la optimización bayesiana es invitado a probar sus propios valores. Solamente tener en cuenta que en el código, la linea
`dataset_grande <- dataset_grande[ foto_mes>=env$data$mes_primero & foto_mes<=env$data$mes_ultimo , ]`
con la finalidad de usar menos memoria esta reduciendo el dataset, eliminar esa linea si se quiere probar una ventana de mas de 12 meses.

Debe quedar claro que se está utilizando la eficiente Optimización Bayesiana para determinar los al mismo tiempo la combinación óptima de parámetros :

- ventana (cantidad de meses donde se entrena)
- arbol rpart
    - maxdepth
    - minsplit
    - minbucket
    - cp

Para el mlrMBO esto es optimizar cinco hiperparámetros al mismo tiempo; mlrMBO no tiene idea que un parámetro regula la cantidad de meses del dataset de training y los otros cuatro el algoritmo rpart.

Plataforma de corrida : Google Cloud

Prerrequisitos

- En el bucket de Google Cloud debe existir la carpeta `cloud1` que es donde estan todos los archivos
- Dentro de la carpeta `cloud1` debe estar las carpetas
    - `R`
    - `datasetsOri`
    - `datasets`
    - `work`

- En el bucket de Google Cloud debe existir los archivos
    - `/cloud1/datasets/paquete_premium_dias.txt`

- Bajar del dropbox de la materia el archivo `R/rpart/rpart_tune_MBO_meses_undersampling.r` a la PC local
- Subir el archivo recien bajado al bucket de Google Cloud a la carpeta `cloud1/R/rpart`

- En la nube debe tener instaladas las librerias de R :
    - `data.table`
    - `rpart`
    - `ROCR`
    - `DiceKriging`
    - `mlrMBO`

Plataforma de ejecución: Cloud, Virtual Machine mortal preemptive , 1 vcpu, 24 GB RAM
Para crear la máquina virtual seguir los pasos del documento ProcesamientoCloud.pdf página 30, dicho documento se encuentra en el dropbox en la carpeta cloud y es el que se utilizó para configurar Google Cloud .

La salida de esta corrida quedará en el archivo `work/hiperparámetro_1501.txt`

3. Correr R/ranger/ranger_tune_MBO_meses_undersampling.r

Esto es lo mismo que se corrió en la tarea 2 para rpart, pero ahora con el algoritmo Random Forest en su versión de la librería ranger.

Prerrequisitos adicionales a los de la Tarea 2

- Bajar del dropbox de la materia el archivo `R/ranger/ranger_tune_MBO_meses_undersampling.r` a la PC local
- Subir el archivo recien bajado al bucket de Google Cloud a la carpeta `cloud1/R/ranger`

- En la nube debe tener instaladas las librerias de R :
  - `data.table`
  - `ranger`
  - `randomForest`
  - `ROCR`
  - `DiceKriging`
  - `mlrMBO`

Plataforma de ejecución: Cloud, Virtual Machine mortal preemptive , 8 vcpu, 48 GB RAM
Para crear la máquina virtual seguir los pasos del documento ProcesamientoCloud.pdf página 30, dicho documento se encuentra en el dropbox en la carpeta cloud y es el que se utilizó para configurar Google Cloud .

La salida de esta corrida quedará en el archivo `work/hiperparámetro_2501.txt`

4. Correr R/xgboost/xgboost_tune_MBO_meses_undersampling.r

Esto es lo mismo que se corrió en la tarea 2 para rpart, pero ahora con el algoritmo XGBoost (eXtreme Gradient Boosting) con la librería homónima.

Prerrequisitos adicionales a los de la Tarea 2

- Bajar del dropbox de la materia el archivo R/xgboost/xgboost_tune_MBO_meses_undersampling.r a la PC local
- Subir el archivo recien bajado al bucket de Google Cloud a la carpeta cloud1/R/xgboost

- En la nube debe tener instaladas las librerias de R :
  - data.table
  - ROCR
  - Matrix
  - xgboost
  - DiceKriging
  - mlrMBO

Plataforma de ejecución: Cloud, Virtual Machine mortal preemptive , 8 vcpu, 48 GB RAM
Para crear la máquina virtual seguir los pasos del documento ProcesamientoCloud.pdf página 30, dicho documento se encuentra en el dropbox en la carpeta cloud y es el que se utilizó para configurar Google Cloud .

La salida de esta corrida quedará en el archivo work/hiperparámetro_3501.txt

5. Correr R/lightgbm/lightgbm_tune_MBO_meses_undersampling.r

Esto es lo mismo que se corrió en la tarea 2 para rpart, pero ahora con el algoritmo LightGBM (Light Gradient Boosting Machine) con la librería homónima.

Prerrequisitos adicionales a los de la Tarea 2

- Bajar del dropbox de la materia el archivo `R/lightgbm/lightgbm_tune_MBO_meses_undersampling.r` a la PC local
- Subir el archivo recien bajado al bucket de Google Cloud a la carpeta `cloud1/R/lightgbm`

- En la nube debe tener instaladas las librerias de R :
  - `data.table`
  - `ROCR`
  - `Matrix`
  - `lightgbm`
  - `DiceKriging`
  - `mlrMBO`

Plataforma de ejecución: Cloud, Virtual Machine mortal preemptive , 8 vcpu, 24 GB RAM
Para crear la máquina virtual seguir los pasos del documento ProcesamientoCloud.pdf página 30, dicho documento se encuentra en el dropbox en la carpeta cloud y es el que se utilizó para configurar Google Cloud .

La salida de esta corrida quedará en el archivo `work/hiperparámetro_4501.txt`

6. Correr R/lightgbm/lightgbm_tune_MBO_meses.r

Esto es lo mismo que se corrió en la tarea 2 para rpart, pero ahora con el algoritmo LightGBM (Light Gradient Boosting Machine) con la librería homónima.

Prerrequisitos adicionales a los de la Tarea 2

Bajar del dropbox de la materia el archivo `R/lightgbm/lightgbm_tune_MBO_meses.r` a la PC local
Subir el archivo recien bajado al bucket de Google Cloud a la carpeta `cloud1/R/lightgbm`

En la nube debe tener instaladas las librerias de R :

- `data.table`
- `ROCR`
- `Matrix`
- `lightgbm`
- `DiceKriging`
- `mlrMBO`

Plataforma de ejecución: Cloud, Virtual Machine mortal preemptive , 8 vcpu, 48 GB RAM
Para crear la máquina virtual seguir los pasos del documento ProcesamientoCloud.pdf página 30, dicho documento se encuentra en el dropbox en la carpeta cloud y es el que se utilizó para configurar Google Cloud .

La salida de esta corrida quedará en el archivo `work/hiperparámetro_4601.txt`

7. Consolidación de las corridas

Hemos corrido cuatro algoritmos, y los parámetros de cada uno (que son distintos) fueron optimizados por nuestra fiel Bayesian Optimization . Además del los parámetros de cada algoritmo hemos optimizado el parámetro ventana que es la cantidad de meses del pasado en los que entrenamos.

Luego de haber corrido las tareas 2,3,4,5 y 6 completar la siguiente planilla y sacar las propias conclusiones .
Atención, para calcular la ganancia esperada en `201904_dias.txt` se debe ordenar la salida por metrica1_actual descendente y luego fijarse el primer valor de metrica1_futuro que queda en la primera linea.
Para calcular el costo total de la corrida en USD debe tener en cuenta el costo por hora de la máquina virtual que creó y la cantidad total de horas que esa máquina virtual estuvo corriendo, sumando todas las veces que debió volver a crear una debido que que Google le mató la máquina debido a que era preemptive . Antes

Tarea	Algoritmo	Negativos	Horas Corrida	Costo total corrida en USD	Meses ventana optima	Ganancia esperada en 201904_dias
2	**CART**	10%				
3	**Random Forest**	10%				
4	**XGBoost**	10%				
5	**LightGBM**	10%				
6	**LightGBM**	100%				

Mirando la tabla anterior completa, qué algoritmo le conviene correr y sobre que dataset ?