Universidad de Buenos Aires
Maestría en Data Mining
DM EyF ciclo 2019

Tareas para el Hogar

Estamos lanzados a la carrera de mejorar la ganancia de nuestro modelo predictivo.
En la clase del jueves 19 de septiembre vimos que tenemos a nuestra disposición la herramienta de la Optimización Bayesiana a la que le pasamos un algoritmo, dataset y sus hiperparámetros a optimizar y luego de algunas horas tenemos la respuesta de cuales son los mejores hiperparámetros para ese algoritmo con ese dataset y clase.

Para mejorar la ganancia estaremos viendo en las clases que restan de la materia :

- The Art of Feature Engineering
  - Pasar fechas absolutas a relativas
  - Tratamiento de nulos, outliers, etc
  - Creación de atributos dentro del mismo mes
  - Creación de atributos históricos
  - Deflacionar atributos monetarios ?

- The Art of Class Engineering
  - Trabajar con una clase binaria que sea la union de {BAJA+1,BAJA+2}
  - Trabajar con bajas del futuro cercano {BAJA+3, BAJA+4}

- Estrategia de entrenamiento
  - Entrenar en la union de varios meses del pasado
  - Testear en meses del futuro y dejar hacer Montecarlo Estimation dentro de un mismo mes.

- Algoritmos mas poderosos que los arboles de decisión
  - Random Forest
  - Gradient Boosting
    - XGBoost
    - LightGBM

De aquí en adelante haremos todas las corridas importantes en la nube de Google Cloud, esta tarea para el hogar será el primer acercamiento formal a la nube. Se sugiere a los alumnos pedir ayuda a sus compañeros con experiencia en IT para estas primeras corridas en la nube.
Debe tenerse en cuenta que la nube solo la estamos utilizando para procesar. A los scripts los

modificaremos siempre en la PC local y luego los subiremos al bucket de Google Cloud para correrlos desde el Rstudio en una máquina virtual.
El bucket siempre está disponible, siempre podemos consultar archivos, subir archivos o bajarlos.
Las máquinas virtuales son muy costosas, con lo cual las creamos y encendemos justo antes de procesar, ingresamos al RStudio, ponemos a correr el script R, monitoreamos el avance, y cuando estamos seguros que ha terminado, apagamos y borramos la máquina virtual. Es muy grave olvidarse la máquina virtual encendida, ya que agotará rapidamente el credito de USD 300 .

Basicamente procesar en la nube consiste de los siguientes pasos :

1. Verificar la existencia en el bucket de Google Cloud de los archivos que serán usados como entrada del proceso, esto se hace en https://console.cloud.google.com/storage/browser/
2. Bajar del dropbox a la PC local los scripts R que se utilizarán, los que generalmente fueron actualizados en el dropbox por el profesor hace pocas horas.
3. De la PC local subir al bucket de Google Cloud los scripts R , desde https://console.cloud.google.com/storage/browser/
4. Crear la máquina virtual con las especificaciones de memoria RAM, cantidad de virtual CPUs, si es mortal o inmortal, etc que hace falta para la corrida de ese script R, desde https://console.cloud.google.com/compute/instances
5. Ingresar al Rstudio web de la máquina virtual
6. Navegar por las carpetas hasta encontrar el script que se desea correr
7. Poner a correr el script
8. Desde el bucket de Google Cloud verificar periodicamente las salidas que va generando el script
9. Una vez que termina el proceso, apagar la máquina vitual y elminarla.
10. Bajarse los resultados de la corrida de la carpeta `work` del bucket de Google Cloud a la PC local
11. Analizar los resultados de la corrida

12. Pensar nuevos experimentos para mejorar la ganancia
13. Hacer cambios a los scripts existentes para el nuevo experimento
14. Ir al punto 1.

1. Tareas de housekeeping

    - Se han hecho cambios a la carpeta R del dropbox, bajarla nuevamente a la PC local

2. Correr `R\rpart\rpart_canarito.r`

    Plataforma de corrida : PC

Lo que sigue es la primer corrida del "Arbol que no tiene hiperparámetros, no hace overfitting ni underfitting, en donde no hace falta hacer training/testing, ni tampoco Optimización Bayesiana" .

En la clase del jueves 19 de septiembre se vieron los resultados de las corridas para encontrar los hiperparámetros óptimos del arbol de decisión rpart con dos métodos que buscan lo mismo : el brutal Grid Search y la elegante Bayesian Optimization, trabajando sobre los archivos `201902_dias.txt` y `201904_dias.txt`
Los archivos `_dias.txt` son aquellos en los que se ha pasado la fecha absolutas a fechas relativas, es decir que ahora las fechas están expresadas en dias.
La optimización Grid Search llevó interminables 35 horas de corrida, la optimización Bayesian Optimization apenas algo más de 1 hora. Está claro que Grid Search se mostró solo a fines pedagógicos, a partir de este momento nunca más en sus vidas de mineros la utilizarán.

Levantar a una planilla de cálculo la salida de Bayesian Optimization (lo generado por `rpart_tune_MBO_01.r` que queda en `work\hiperparametro_1200.txt`) y ordernarla por `metrica1_actual` descendente (que es la ganancia promedio calculada en `201902_dias.txt` haciendo 5-fold Montecarlo). Tomar nota de cuales son los mejores hiperparámetros y cual es el valor de `metrica1_futuro` , esa es la ganancia que tiene el mejor modelo en 201904 que podemos lograr usando rpart entrenado en `201902_dias.txt` y aplicandolo a `201904_dias.txt`

Es fundamental ordenar por `metrica1_actual` , jamas se debe elegir los mejores hiperparámetros ordenando por `metrica1_futuro` ya que ese dato es del futuro y no va a estar disponible en los datos de junio 201906 para la entrega final. Visto de otra forma, consideren `metrica1_futuro` la ganancia que obtendrían en la competencia de la materia.

rpart actual 201902_dias.txt futuro 201904_dias.txt			
Método	**Ganancia Futura**	**AUC Futuro**	**Tiempo (minutos)**
Grid Search	5,971,500	0.8456	1800
Bayesian Optimization	6,109,000	0.8411	90
Canary Attributes ™	6,119,500	0.8565	1
Canary Attributes Weight ™®	6,329,500	0.8656	1

En principio los tres siguientes métodos deben encontrar aproximadamente el mismo set de hiperparámetros optimos y producir aproximadamente la misma ganancia máxima, pero en tiempos muy distintos.
La diferencia es que el método de los canaritos es muchísimo mas rápido que los dos anteriores porque no tiene que buscar hiperparámetros optimos.

Los pasos seguidos en el código son :

1. Se le agregan al dataset un 20% mas de atributos nuevos, los canaritos, que son variables random con distribución uniforme en el intervalo (0,1) . En nuestro dataset son 34 atributos canaritos dato que tenemos 169 + 1 atributos
2. Por un detalle del orden en que rpart recorre los atributos cuando elige la major variable y el mejor corte, las variables canarito se agregan al comienzo del dataset
3. Se genera un arbol con la máxima profundidad que permite rpart (30) dejando que siempre se abra (cp=0) , pero debido a un problema que tiene rpart se establece minbucket=1
4. Una vez generado el arbol, con un hacking a la salida del arbol, se hace el prunning de todo lo que cuelga de un atributo canarito
5. Finalmente, se miden las ganancias y AUCs de el arbol original y el arbol podado en los canaritos.

Correr `R\rpart\rpart_canarito.r` linea a linea, viendo los valores que van tomando las variables y fijarse cuanta ganancia se obtiene, prestar atencion a los archivos que quedan en la carpeta `work`

- `rpart_canarito.jpg`
- `rpart_canarito_pruned.jpg`

En primer lugar, lo que se está haciendo es ineficiente, ya que se deja crecer el arbol primero, y luego se lo poda donde estan los canaritos, el algoritmo NO deberia hacer el split de un nodo si el mejor corte es por un canarito
En segunda instancia, y más importante aún, en teoría no hacen falta los nuevos atributos canarito para decidir cuando no seguir creciendo el arbol, ya que se podría hacer un test estadístico que tuviera en cuenta la cantidad de variables disponibles para hacer el split, los tipos de estas variables, y

la cantidad de valores , pero esta idea esta aún bajo exploración.

Finalmente, el objetivo NO es el arbol de decision, sino gradient boosting, XGBoost y LightGBM, lograr que esos algoritmos no tengan hiperparámetros, incluso que la cantidad de arboles no sea un hiperparámetro, sino que cuando al intentar generar el arbol n+1 si el mejor corte en la raiz es un canarito, entonces ya no tiene sentido seguir agregando arboles al ensemble y finaliza el boosting.

Un gran avance en XGBoost y LightGBM es la regularización. Dado un nodo, si con una cantidad de pos y neg , la forma naif de estimar la probabilidad en nuevos datos en pos/(pos+neg) pero sabemos que por la forma en que se buscó, ese número es optimista.
La forma en que se regulariza (nomenclatura XGBoost) es prob_pos_esperada = (pos $\pm \alpha$) / (pos+neg + λ) donde α y λ son hiperparámetros que se optimizan con una Bayesian Optimization. Es posible evitar una búsqueda y resolverlo con una idea similar a la de los canaritos ?

3. Correr R/rpart/rpart_tune_MBO_02.r

   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/dias/201812_dias.txt`
     - `/cloud1/datasets/dias/201902_dias.txt`
     - `/cloud1/datasets/dias/201904_dias.txt`

   - Bajar del dropbox de la materia el archivo `R/rpart/rpart_tune_MBO_02.r` a la PC local
   - Subir el archivo recien bajado al bucket de Google Cloud a la carpeta `cloud1/R/rpart`

   Plataforma de ejecución: Cloud, Virtual Machine mortal preemptive , 1 vcpu, 8 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_1210.txt`

4. Correr `R/FeatureEngineering/fe_presente.r`

   Plataforma de corrida : Google Cloud

   En clase se verá el objetivo de este programa en gran detalle. Por favor correrlo para ya tener la salida generada al comienzo de la clase.

   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 el archivo `/cloud1/datasetsOri/paquete_premium.zip`
   - Bajar del dropbox de la materia el archivo `R/FeatrureEngeneering/fe_presente.r` a la PC local
   - Subir el archivo recien bajado al bucket de Google Cloud a la carpeta `cloud1/R/FeatureEngineering`

   Plataforma de ejecución: Cloud, Virtual Machine inmortal pagando el precio fulll, ya que este proceso no está programado para retomar el procesamiento en caso de apagarse la máquina.

   8 vcpu, 48 GB RAM

   Este proceso demora aproximadamente 20 minutos, por favor recordar apagar la maquina virtual al finalizar ya que esta es inmortal y **no** se apaga automáticamente a las 24 horas. Una vez apagada es conveniente borrarla.

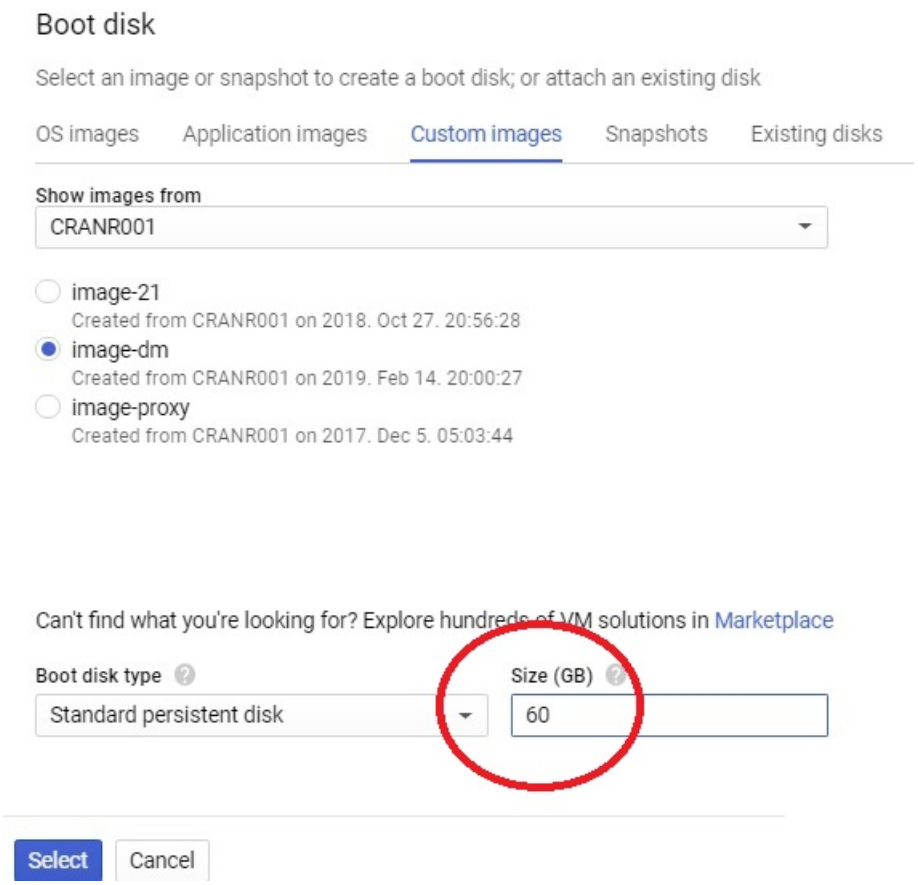
   La salida de este programa consiste en 36 datasets que se crearán en la carpeta `datasets/ext` `más el archivo datasets/paquete_premium_ext.txt.gz`

5. Correr en la nube `/FeatureEngineering/fe_historia.r`

En clase se verá el objetivo de este programa en gran detalle. Por favor correrlo para ya tener la salida generada al comiendo de la clase.

Plataforma de ejecución: Cloud, Virtual Machine inmortal pagando el precio fulll, ya que este proceso no está programado para retomar el procesamiento en caso de apagarse la máquina.

8 vcpu, 52 GB RAM , al elegir la imagen del sistema operativo asignar 60GB al espacio en disco.



Este proceso demora aproximadamente 2 horas 20 minutos, por favor recordar apagar la maquina virtual al finalizar ya que esta es inmortal y **no** se apaga automáticamente a las 24 horas. Una vez apagada es conveniente borrarla.