Trabajo Práctico 3: Model Selection

Fridman Axel	Hsueh Noé	Salas Héctor
527/20	546/19	Postgrado

Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Machine Learning Práctico

Objetivos

En el presente informe presentaremos un Business Case hipotético, para el cual buscaremos una solución aplicando técnicas de Machine Learning. Comenzaremos haciendo un repaso del Business Case planteado en el TP1, con su respectiva solución general (TP 2), luego probaremos una variedad de modelos (Baseline propio, Regresión Logística, XGBoost, LightGBM, Gradient Boosting, Random Forest y SVM) y analizaremos diferentes formas de imputar los datos faltantes. A partir de estos modelos, elegiremos uno según la métrica log loss y en consideración a la capacidad de distinguir las clases.

Se adjunta junto al informe un notebook con el código usado para la selección de modelos.

Business Case

Una clínica quiere reducir las muertes de sus pacientes por ACV: se requiere de la identificación de los pacientes de alto riesgo para poder brindarles, a sus respectivos médicos de cabecera, una notificación temprana del caso en pos de tomar las medidas de prevención necesarias. Nuestro KPI será la reducción de las muertes por accidentes cardiovasculares anuales en un 20 % en un lapso de 5 años en nuestro hospital.

Solución

La solución a este problema es la implementación de un software predictivo vinculado con la base de datos de cada paciente en pos de identificar el grado de riesgo que tienen los pacientes de padecer un ACV. Luego, esto permitiría, en conjunto con el médico (quien posee en este caso conocimiento del dominio), recomendar un tratamiento y las precauciones necesarias para prevenir el ACV.

Hipótesis preliminares

Asumiremos que los datos que obtuvimos son una muestra representativa de la población de la que queremos inferir. A continuación, nos enfocaremos en la selección de un modelo de Machine Learning para la solución a nuestro problema. Cabe destacar que el modelo constituye una parte de la solución general y no es la solución al problema, dado que para ello se necesitaría de, por ejemplo, conocimiento del dominio. Asimismo, tampoco estamos teniendo en cuenta la validación de dicho modelo en la vida real; por lo que nuestro resultado final cuenta con las limitaciones propias del dataset.

0. Introducción

0.1. Preparación de datos

Luego de una extensiva exploración de datos, incluyendo el análisis realizado en los trabajos prácticos anteriores, identificamos ciertas particularidades en nuestro dataset,

que describiremos a continuación. En primer lugar, notamos que había un número considerable de valores faltantes en la columna de BMI (Índice de masa corporal), aproximadamente el mismo número de valores pertenecientes a la clase minoritaria. La distribución del BMI de acuerdo a la variable que buscamos predecir (stroke, pacientes positivos y negativos al ACV) se muestra a continuación.

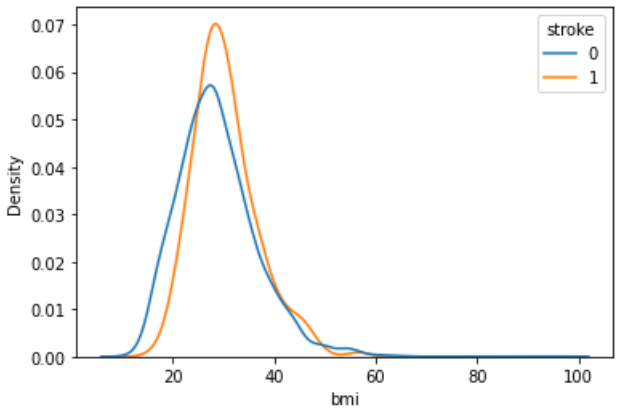

Figura 1: Imagen de densidades de BMI según estado ACV.

Se observó que las distribuciones cercanas a la moda son similares cuando se trata de BMI muy bajos (entre 15 y 20), y la densidad de personas con ACV se hace más pequeña en comparación a la densidad de aquellos sin ACV. También notamos una baja correlación lineal entre BMI y el resto de las variables individualmente (Tabla 1). En consecuencia, decidimos imputar los valores faltantes de la columna BMI mediante varios procedimientos, como el uso de una regresión lineal entrenada con los valores no nulos de las demás variables.

Variable	Correlacion con BMI
Age (Edad, número entero)	0.333314
Hypertension (hipertensión, binario tiene o no)	0.167770
Heart disease (enfermedad cardíaca, binario tiene o no)	0.041322
Avg_glucose_level (Nivel de glucosa promedio, número real >= 0)	0.175672
Stroke (ACV Variable a predecir, binario, tuvo o no)	0.042341

Tabla 1: Correlaciones de BMI con otras variables numéricas del conjunto de datos.

Por otro lado, observamos que existía una única observación en la variable género con la categoría “other” (otro). Representaba el 0,02 % de la muestra y los valores de las demás variables para dicha observación eran similares a los observados en otras instancias del género (Masculino o Femenino), por lo que decidimos descartarla.

Además, observamos que en la columna de valores categóricos “smoking status” (estado en cuanto a fumar), la categoría “Unknown” (se desconoce si fuma o no) representa el 30.22 % del total de observaciones (Tabla 2). Esta categoría también posee una propensión más baja a padecer ACV (figura 2).

Estado en cuanto a fumar	Cantidad	Porcentaje sobre total
never smoked (Nunca fumó)	1892	37.03 %
Unknown (No se sabe nada)	1544	30.22 %
formerly smoked (Fumó y dejó)	884	17.30 %
Smokes (Fuma actualmente)	789	15.44 %

Tabla 2: Valores de cada categoría de la variable smoking status.

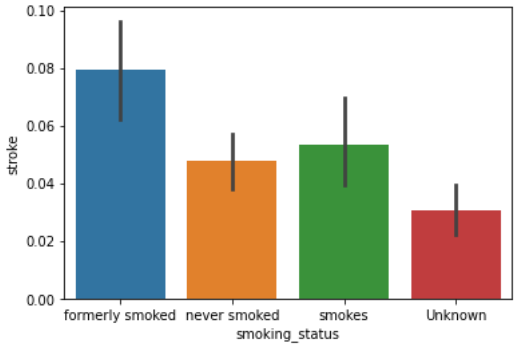

Figura 2: Número de pacientes con ACV en función del estado de fumar.

Sin embargo, es probable que su distribución no sea estadísticamente diferente respecto a la distribución del grupo de pacientes que fuman y a la del grupo de pacientes que nunca fumó, pero probablemente si es diferente a la de aquellos que fumaron y luego dejaron de fumar. En consecuencia, basados en las observaciones mencionadas previamente sobre los datos, decidimos aplicar tres tipos de preprocesamientos (preprocesamiento 1, 2 y 3), tomando en cuenta operaciones de escalamiento (normalización), creación de variables dummy e imputación. La normalización de las variables numéricas facilita la identificación de características por parte del modelo. En modelos de alta interpretabilidad, como en la regresión lineal, permite comparar las variables de acuerdo a la influencia que tienen sobre la variable respuesta, ya que de esa esa forma no presentan diferentes rangos de valores (escala y dispersión). Las variables dummy permiten que este tipo de modelos pueda utilizar la información contenida en las variables categóricas y la imputación permite aprovechar las observaciones que tienen datos faltantes.

Antes de aplicar estas operaciones, eliminamos algunas variables (`id`, `work_type`, `residence_type`, `ever_married`) del conjunto de datos debido a la influencia poco clara sobre la variable stroke.

0.1.1. Preprocesamiento 1

Este pipeline de preprocesamiento incluye tres operaciones sucesivas: imputación de la variable BMI con el valor promedio de todos sus valores (Simple Imputer), normalización de las variables numéricas `age`, `hypertension`, `avg_glucose_level`, y la creación de variables dummy para las columnas `gender`, `ever_married`, `smoking_status`, `heart_disease`. En la variable `smoking_status` se reconoce la categoría “Unknown” como una categoría más.

0.1.2. Preprocesamiento 2

En este preprocesamiento se establecen los valores de la categoría "Unknown" en la variable `smoking_status` como valores faltantes. Luego se aplican las operaciones de escalamiento y creación de variables dummy sobre las mismas variables del preprocesamiento 1 y finalmente, se aplica el imputador iterativo (Iterative Imputer). El algoritmo aplica una regresión lineal a una variable con valores faltantes y predice dichos valores utilizando el resto de las variables. Este proceso se realiza iterativamente hasta predecir los valores faltantes en todas las variables que los tengan.

0.1.3. Preprocesamiento 3

Se aplica inicialmente una regresión lineal a la variable BMI en función de las restantes para imputar los valores faltantes. Luego se establecen los valores de la categoría "Unknown" en la variable `smoking_status` como valores faltantes y se aplica un árbol de decisión para predecir sus valores. Finalmente se normalizan las variables numéricas y se crean las variables dummy como en los otros pipelines de preprocesamiento.

0.2. Partición de datos

Luego de explorar todo el dataset y haber visto cuál era el estado de los datos que teníamos, decidimos particionar el dataset de manera estratificada por número de personas con ACV (nuestra variable a predecir). El 80 % del dataset corresponderá a entrenamiento, el 10 % a desarrollo, y 10 % restante a la evaluación de nuestro modelo. De manera que en cada uno de estos tres datasets, todas las transformaciones o imputaciones son dadas con información de ese mismo dataset y sin usar los otros dos. Es decir, cuando se quiere imputar BMI en data train solo se usa como modelo para imputar a los datos del mismo, evitando de esta forma data leakage.

0.3. Métrica offline

A lo largo del trabajo hicimos pruebas con varias métricas. El primer intento consistió en una métrica personalizada que penaliza en mayor medida las predicciones de probabilidades mucho menores que 1 en las personas que tuvieron ACV. Al mismo tiempo, tenía un peso relativamente pequeño para aquellas predicciones cercanas al cero cuando la persona no tuvo ACV. Definimos el parámetro K para representar el costo que tiene informar erróneamente que un paciente no está enfermo de gravedad cuando sí lo está ($K > 1$), y el que existe al informar que lo está, cuando en realidad se encuentra sano ($K = 1$). Por ejemplo,

$$\text{Costo}(x_i, y_i) = \begin{cases} K & \text{si } (y_i = 1 \wedge x_i \neq y_i) \\ 1 & \text{si } (y_i = 0 \wedge x_i \neq y_i) \\ 0 & \text{si cc} \end{cases}$$

No obstante, esta métrica era difícil de interpretar y se necesitaba el conocimiento de un experto en el dominio para establecer su valor. Por esta razón fue necesario establecer pruebas con distintos valores de K para obtener el valor más adecuado. Cuando $K \to +\infty$,

el modelo con todos los valores cercanos a 1 tenía los mejores resultados en el development set. "Todos tienen ACV a menos que se demuestre con extrema confianza qué no". Cuando $K \to 1$, el modelo con todos los valores en 0 tenía los mejores resultados en el development set. Esto se debe a que, al existir un número reducido de observaciones con ACV, el costo se minimiza; el modelo indica que "Nadie tiene ACV" pero se equivoca un 2 % de las veces (la pifió porque si tenían). Por los motivos citados anteriormente (falta de un coeficiente K confiable sumado a su baja interpretabilidad), decidimos buscar otra métrica alternativa. Considerando que tratamos de resolver un problema de regresión en el que se obtiene un score entre (0,1) indicando la proximidad a la clase que pertenece, nuestra métrica debía representar este score como una probabilidad. Una métrica que cumple con este criterio es la Log-Loss, que fue la que escogimos para comparar la efectividad de los diferentes modelos.

0.4. Calibración de las predicciones del modelo

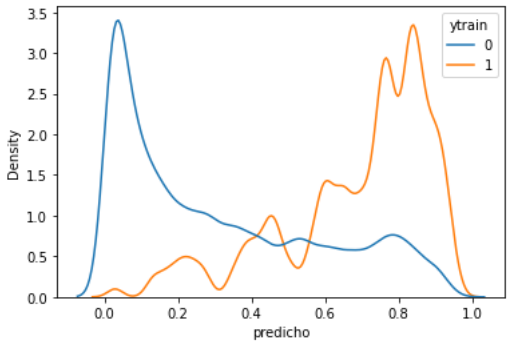

Figura 3: Distribución del score obtenido mediante un modelo de Regresión Logística aplicado a un conjunto de datos de entrenamiento con dos clases (0 y 1). Clase 0: pacientes con bajo riesgo de sufrir un ACV (azul). Clase 1: pacientes con alto riesgo de sufrir un ACV (naranja).

La calibración de un modelo indica que las probabilidades que predice para cada clase deben ser iguales a la frecuencia relativa de observaciones que pertenecen a esa clase, aunque esto será más o menos cierto dependiendo de la cantidad de observaciones que tenga la muestra. Esta es una operación que debe aplicarse en la salida de muchos tipos de modelos, ya que puede mejorar las predicciones que producen. En la Figura 3 se observa la distribución de las probabilidades otorgadas a cada una de las dos clases del conjunto de datos por un modelo de Regresión Logística. Aunque las distribuciones están bien separadas, no representan efectivamente las probabilidades de pertenecer a cada clase. Si tomamos las observaciones cuyas probabilidades están en el intervalo [0.19; 0.21], notamos que sólo el 5 % de aquellas observaciones tuvieron ACV; similarmente de todas las observaciones con probabilidades entre [0.79; 0.81] solamente el 20 % tuvo ACV. Es decir, el modelo nos indica que hay varias observaciones con una probabilidad cercana al 20 % de pertenecer a la clase 1, pero la frecuencia relativa de estas observaciones es de solo el 5 %. Y algo similar ocurre con las probabilidades cercanas al 80 %, en donde la frecuencia relativa de las observaciones es de solo el 20 %. Esto nos indica que sistemáticamente las predicciones del modelo no representan adecuadamente las probabilidades de pertenecer

a cada clase. Por tal razón hace falta un proceso de calibración de las probabilidades estimadas.

1. Selección de modelo

A continuación presentamos los modelos a analizar en este trabajo práctico. Las implementaciones que mencionaremos son modelos utilizados para resolver problemas de clasificación que normalmente proporcionan un vector de salida con las etiquetas de las observaciones pertenecientes a cada clase. Sin embargo, debido a la necesidad de predecir el grado de riesgo que tiene un paciente de sufrir un ACV, en el marco de una "escala o índice de propensión", decidimos utilizar las probabilidades que predicen los modelos como medio para representar dicho riesgo.

1.1. Descripción de modelos a comparar

1.1.1. Modelo Baseline

Un conjunto de datos de entrenamiento con cinco covariables se particionan en dos grupos: positivos al ACV (Grupo A) y negativos al ACV (Grupo B). Se toma el valor promedio de cada una de las covariables en ambos conjuntos de datos. Estos valores promedios pueden variar entre ambos conjuntos de datos (e.g. el promedio de edad para el Grupo A es más grande que el valor promedio de la misma covariable en el Grupo B). La respuesta (Score) arrojada por el modelo se construye penalizando los valores de las covariables (características, features) de tal modo que, al estar sus valores más cerca del promedio obtenido para el Grupo A que del Grupo B, se les suma 0.2 unidades. En consecuencia, una observación (paciente) con características muy similares a las del Grupo A tendrá un Score alto, mientras que una persona con características más parecidas a las del Grupo B tendrá un Score bajo.

1.1.2. Modelo 1: Logistic Regression

La razón por la que consideramos este modelo como una primera elección se debe a la simplicidad tanto en implementación como en interpretación. Para este caso tendremos tres posibles elecciones, dependiendo del tratamiento de los datos faltantes para bmi y smoking status:

1. Logistic Regression simple con Simple Imputer para BMI y tomando Unknown del smoking status como una clase adicional
2. Logistic Regression con IterativeImputer de Sklearn
3. Logistic Regression con una regresión lineal para imputar los datos faltantes del BMI y un árbol de decisión para imputar los datos faltantes de smoking status.

1.1.3. Modelo 2: LightGBM

Análogamente al caso anterior, realizaremos tres submodelos distintos para luego compararlos. Los tratamientos son similares al caso anterior:

1. LightGBM simple con Simple Imputer para BMI y tomando Unknown del smoking status como una clase adicional

2. LightGBM con IterativeImputer de Sklearn

3. LightGBM con una regresión lineal para imputar los datos faltantes del BMI y un árbol de decisión para imputar los datos faltantes de smoking status.

1.1.4. Modelo 3: XGBoost

Análogamente al caso anterior, realizaremos tres submodelos distintos según el tratamiento previo:

1. XGBoost simple con Simple Imputer para BMI y tomando Unknown del smoking status como una clase adicional

2. XGBoost con IterativeImputer de Sklearn

3. XGBoost con una regresión lineal para imputar los datos faltantes del BMI y un árbol de decisión para imputar los datos faltantes de smoking status.

1.1.5. Modelo 4: GBM

En este caso también aplicamos los pipelines de preprocesamiento previamente mencionados, pero realizamos un balanceo manual debido a que la implementación de Sklearn de este algoritmo no posee el atributo class weight. Para determinar el efecto de la operación de calibración de las probabilidades, se obtuvieron las salidas de los modelos sin y con el proceso de calibración usando los datos de entrenamiento.

1.1.6. Modelo 5: Random Forest (RF)

Se aplica el mismo procedimiento que en los casos anteriores más la operación de calibración realizada para el modelo GBM.

1.1.7. Modelo 6: SVM

Se compararon tres modelos con los preprocesamientos respectivos en los datos descritos previamente más la operación de calibración.

1.2. Análisis de los resultados

A continuación, se muestra el desempeño de los diferentes modelos con los diferentes pipelines de preprocesamiento con el objetivo de escoger la secuencia más efectiva para el modelo candidato. En la Tabla 3, observamos el desempeño de los diferentes modelos según la métrica Log-Loss. En esta primera iteración observamos que la regresión logística tiene un bajo desempeño, incluso por debajo del modelo Baseline. Por esta razón, decidimos probar con modelos más complejos, como el LightGBM y el XGBoost. Sin embargo, los resultados indican que estos modelos experimentan sobreajuste, lo que probablemente

se debe al bajo número de observaciones y a un ajuste rudimentario (casi al azar) de los hiperparámetros. El sobreajuste es más pronunciado en el modelo LightGBM.

name	train	dev
baseline	0.347787	0.443975
lr con Simple Imputer	0.499793	0.467997
lr con IterativeImputer	0.499680	0.468944
lr con regresion de bmi y DTC en ss	0.497902	0.467450
lgb con Simple Imputer	0.111823	0.249727
lgb con IterativeImputer de bmi	0.156115	0.283055
lgb con regresion de bmi y DTC en ss	0.165008	0.283389
xgb con Simple Imputer	0.374278	0.400972
xgb con IterativeImputer de bmi	0.386384	0.402270
xgb con regresion de bmi y DTC en ss	0.383157	0.407418
gb con Simple Imputer	0.143284	0.165180
gb con IterativeImputer de bmi	0.144547	0.168300
gb con regresion de bmi y DTC en ss	0.145235	0.168865
rf con Simple Imputer	0.067423	0.174196
rf con IterativeImputer de bmi	0.087408	0.186755
rf con regresion de bmi y DTC en ss	0.086706	0.183582
SVM con Simple Imputer	0.106734	0.178024
SVM con IterativeImputer de bmi	0.143667	0.177588
SVM con regresion de bmi y DTC en ss	0.139394	0.181319

Tabla 3: Desempeño de los modelos utilizando la métrica Log-Loss. Para cada tipo de modelo se entrenaron tres modelos con datos modificados de acuerdo a los pipelines de preprocesamiento 1, 2 y 3 respectivamente.

Se observa que el Preprocesamiento 1 resultó ser el pipeline más efectivo de acuerdo al error de entrenamiento para todos los modelos salvo la regresión logística (aunque con muy poca diferencia). El error de validación también resultó más bajo con esta secuencia para la gran mayoría de los modelos. Esto significa que la aplicación de operaciones complejas de imputación no reflejan mejores resultados en las predicciones de los modelos. En consecuencia, decidimos utilizar el pipeline de Preprocesamiento 1 durante el proceso de selección de hiperparámetros para los modelos candidatos que describiremos próximamente. El ajuste de los hiperparámetros ayuda a reducir el grado de sobreajuste de un modelo candidato, por lo que se espera que después de este proceso los errores de entrenamiento y validación sean más parecidos. Hay que notar que los modelos de GB, RF y SVM fueron calibrados mediante el paquete de calibración ofrecido por la librería scikit-learn, utilizando el método "Isotónico" con el atributo de validación cruzada establecido en 10. Este tratamiento mejora el desempeño (ver sección de calibración más abajo y para más detalles, ver la notebook), lo que explica que tengan un desempeño mejor. Sin embargo, remarcamos que el objetivo de los análisis de esta sección fue determinar la secuencia de preprocesamiento más efectiva para cada tipo de modelo.

1.3. Selección de hiperparámetros

El proceso de selección de hiperparámetros se llevó a cabo con la librería Hyperopt. El objetivo de esta sección fue seleccionar el modelo con la menor Log-Loss y con el nivel más bajo posible de sobreajuste. Por razones de simplicidad y optimización, así como para no extender el documento de forma excesiva, decidimos aplicar el proceso únicamente a dos de los modelos que probamos previamente. Escogimos la Regresión Logística por su simplicidad y el XGBoost por la eficiencia de la implementación en scikit-learn, que fue la librería utilizada. Los análisis para los demás algoritmos son análogos, y se pueden observar en la notebook del documento. Se utilizó el método Tree-structured Parzen Estimators (TPE) para seleccionar los parámetros.

1.3.1. Selección de hiperparámetros para LightGBM

Los hiperparámetros que consideramos ajustar son los siguientes: `num_leaves`, `n_estimators`, `subsample`, `learning_rate`, `reg_alpha`. Luego se realizaron cien corridas con Hyperopt obteniendo los resultados que se muestran en la Figura 4.

En el gráfico notamos que los mejores modelos encontrados con Hyperopt continúan presentando algo de sobreajuste, dado que el error de entrenamiento (naranja) es más generalmente más bajo en relación al error de validación (azul).

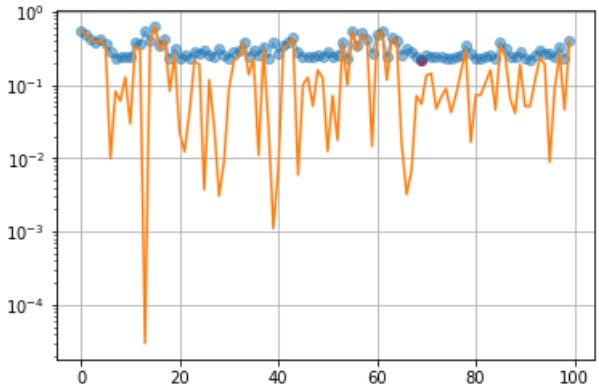

Figura 4: Error de entrenamiento (naranja) y validación (azul) obtenido para cada uno de los 100 modelos de LightGBM probados con el algoritmo de Hyperopt.

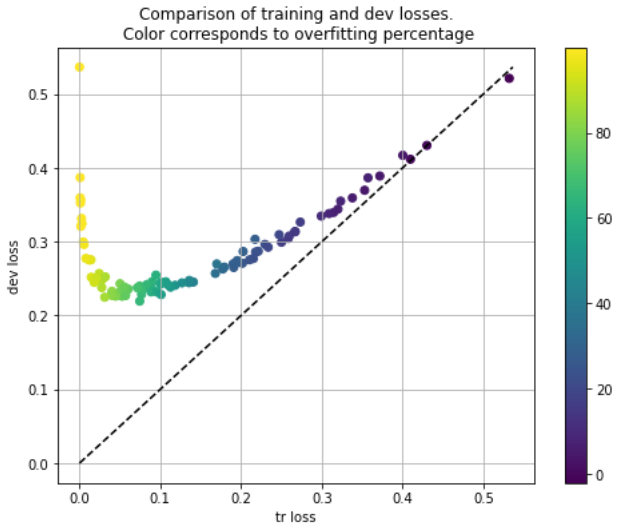

Figura 5: Comparación de los errores de entrenamiento y validación obtenidos para cada uno de los 100 modelos de LightGBM probados con el algoritmo de Hyperopt. Los colores corresponden al porcentaje de sobreajuste.

En la Figura 5, notamos que los modelos que no sobreajustan tienen un desempeño similar al XGBoost (con parámetros no ajustados) y al modelo Baseline (`dev_loss` al 0.3 y 0.4). Por lo tanto, es probable que al ajustar los parámetros del XGBoost con este procedimiento, encontremos un modelo que supere al mejor modelo de LightGBM obtenido.

1.3.2. Selección de hiperparámetros para XGBoost

Realizamos un procedimiento similar al anterior con XGBoost. En este caso elegimos los siguientes hiperparámetros para optimizar: `n_estimators`, `max_depth`, `learning_rate`, `gamma`, `subsample`, `reg_alpha`, `scale_pos_weight`; y al igual que en caso anterior, se realizaron 100 corridas utilizando el método TPE de Hyperopt.

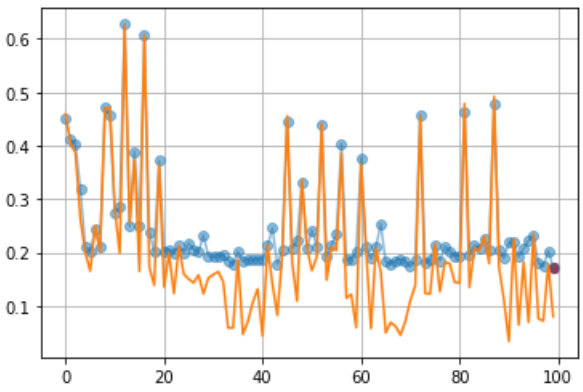

Figura 6: Error de entrenamiento (naranja) y validación (azul) obtenido para cada uno de los 100 modelos probados con el algoritmo de Hyperopt.

A diferencia de lo que observamos con LightGBM, los errores de entrenamiento y validación son mucho más similares (Figura 6).

En la Figura 7 se observa que los modelos con poco porcentaje de sobreajuste tienen un error de validación cercano al 0.2. Por lo tanto es posible seleccionar un modelo con mejor desempeño que en el caso del LightGBM. Seleccionamos un modelo con una diferencia baja entre el error de entrenamiento y validación. Con los hiperparámetros optimizados para el XGBoost, pudimos constatar que efectivamente el error (Log-Loss) disminuyó significativamente (de 0.4 a 0.16). Por otro lado, los modelos optimizados de GB, RF y SVM, ofrecieron resultados muy similares al XGBoost. Creemos que las ligeras mejoras observadas en los errores de entrenamiento y/o validación son producto de la variabilidad generada por el bajo número de observaciones existentes para realizar el entrenamiento de los modelos (4). Por lo tanto, decidimos continuar nuestros análisis con el XGBoost, dado que los análisis probablemente serían similares con los modelos de GB, RF o SVM presentados.

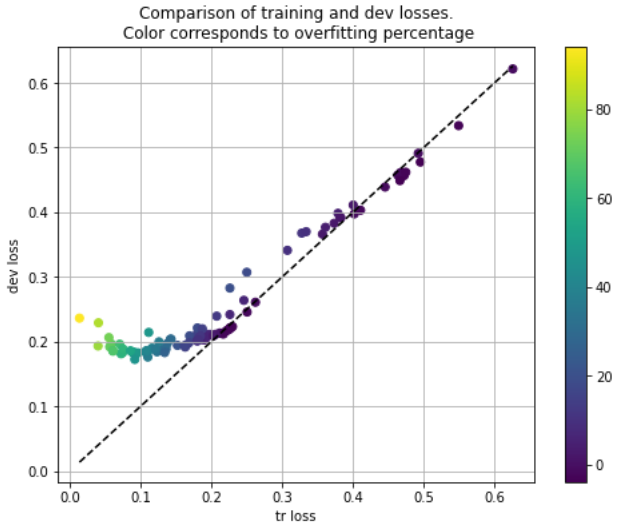

Figura 7: Comparación de los errores de entrenamiento y validación obtenidos para cada uno de los 100 modelos de XGBoost probados con el algoritmo de Hyperopt. Los colores corresponden al porcentaje de sobreajuste.

name	train	dev
baseline	0.347787	0.443975
lr	0.499793	0.467997
lgb	0.111823	0.249727
xgb	0.374278	0.400972
xgb optimizado	0.135736	0.160387
gb optimizado	0.125629	0.158282
rf optimizado	0.132467	0.162518
svm optimizado	0.146842	0.162898

Tabla 4: Valores de entrenamiento y validación de los diferentes tipos de modelos entrenados con datos modificados por el pipeline de Preprocesamiento 1.

Para los casos en donde el modelo falla, veamos si podemos mejorarlos. En la figura 8

se observa la distribución del error absoluto (o sea, $|y - \hat{y}|$), notamos que hay muy pocos ejemplos en donde el modelo tiene un error superior al 70 %.

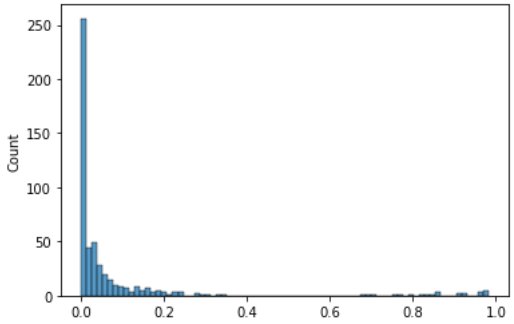

Figura 8: distribución del error absoluto entre lo predicho y lo real

Se dividió los datos entre los mejores ejemplos (errores menores al 0.2) y los peores ejemplos (mayores al 0.7), para muchas variables no se observan que las distribuciones son diferentes salvo para la edad donde la diferencia es más notoria (cf. figura 9). Esto es lógico dado que el ACV ocurre normalmente en personas mayores, por lo que el dataset tendrá muy pocas observaciones con ACV y edades bajas. No obstante, los errores graves representan solo el 0.04 % de los datos del dev. Consideramos que este error difícilmente se pueda mejorar dado el tamaño del dataset.

Por otro lado, nos interesa saber para cada modelo, cómo es la distribución de las predicciones para cada clase. En efecto, un modelo óptimo tendría las distribuciones de las predicciones bien diferenciadas, además de estar cercano a las probabilidades de pertenencia a la clase correspondiente. Esto nos conlleva a la próxima sección donde veremos si se cumple este objetivo o no.

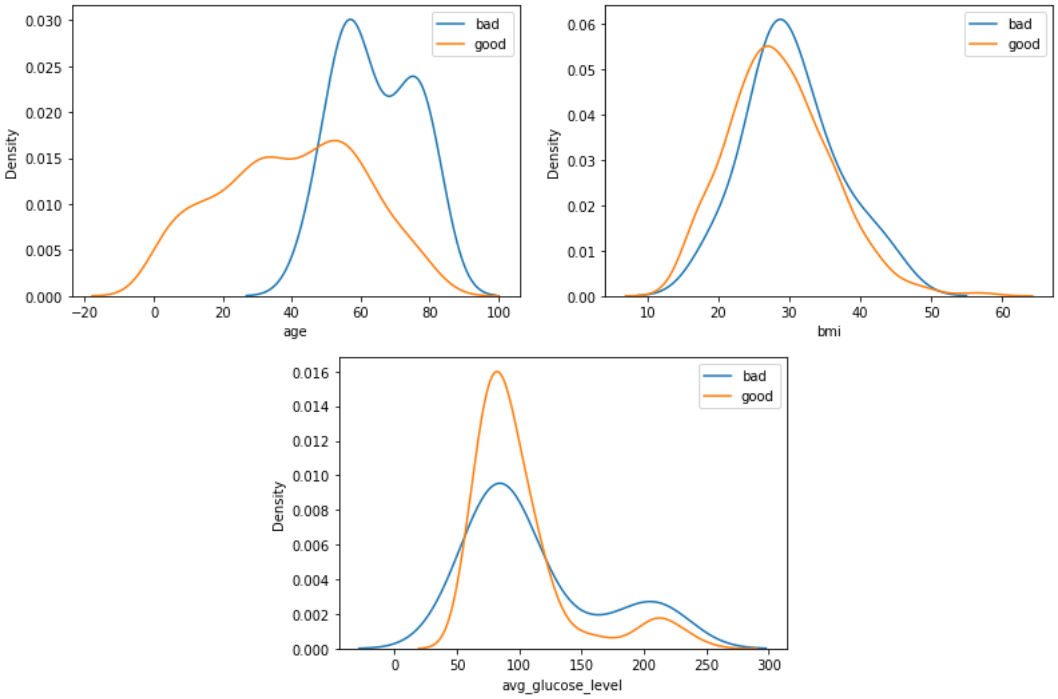

Figura 9: distribución de variables numérica para diferentes ejemplos. En celestes, observaciones con mal error, en naranja observaciones con buen error.

1.4. Calibración de probabilidades

En la figura 10, observamos las distribuciones estimadas para el modelo baseline. Tanto en el train como en el dev, las distribuciones estimadas tienen un alto grado de solapamiento, lo que demuestra que el modelo baseline probablemente puede mejorarse significativamente

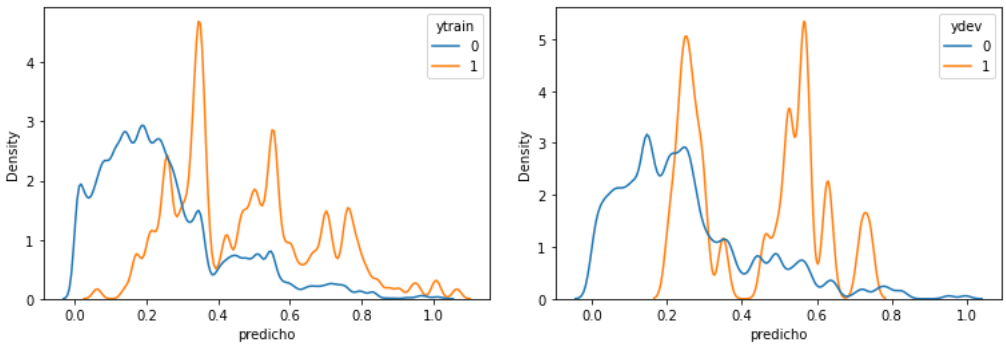

Figura 10: distribución de probabilidades para cada una de las clases utilizando los datos de entrenamiento y validación con el modelo baseline. Los pacientes con bajo riesgo de sufrir un ACV se muestran en azul, mientras que los de alto riesgo en naranja. Izquierda: datos de entrenamiento. Derecha: Datos de validación.

Para la regresión logística, los resultados no son tan optimistas en dev, ya que los separa un poco peor, lo bueno de este tipo de gráfico es que nos permiten detectar el overfitting, que veremos en el siguiente caso.

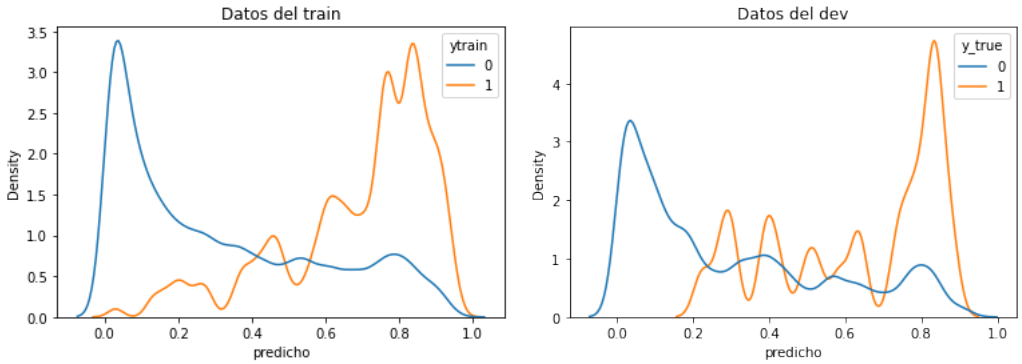

Figura 11: distribución de probabilidades para cada una de las clases utilizando los datos de entrenamiento y validación con el modelo logistic regression. Los pacientes con bajo riesgo de sufrir un ACV se muestran en azul, mientras que los de alto riesgo en naranja. Izquierda: datos de entrenamiento. Derecha: Datos de validación.

Para el caso de LightGBM (cf. figura 12), notamos que en el train las distribuciones son bien separadas pero en el dev el modelo ya no logra diferenciar tan bien entre las distintas clases. Esto se debe al overfitting del modelo descrito anteriormente.

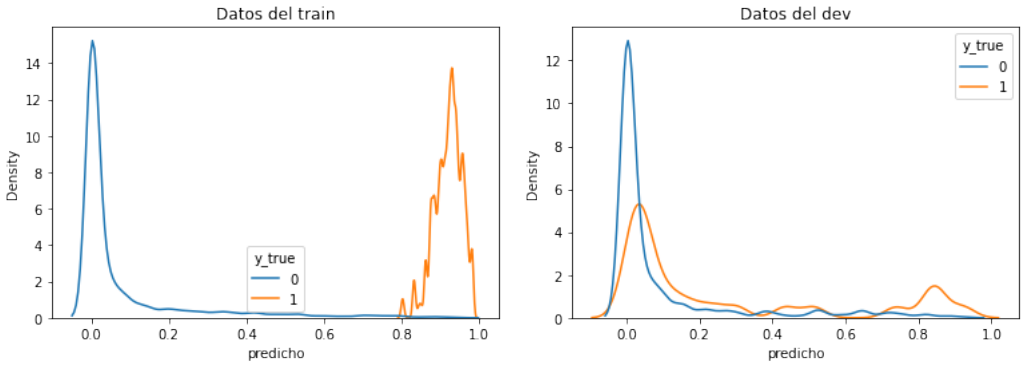

Figura 12: distribución de probabilidades para cada una de las clases utilizando los datos de entrenamiento y validación con el modelo LightGBM. Los pacientes con bajo riesgo de sufrir un ACV se muestran en azul, mientras que los de alto riesgo en naranja. Izquierda: datos de entrenamiento. Derecha: Datos de validación.

Para el caso de XGBoost (cf. figura 13), notamos que en el train las distribuciones son bien separadas, en el dev, a diferencia del LightGBM, se conserva esta separación. Esto es lo que permite que el modelo pueda diferenciar las clases.

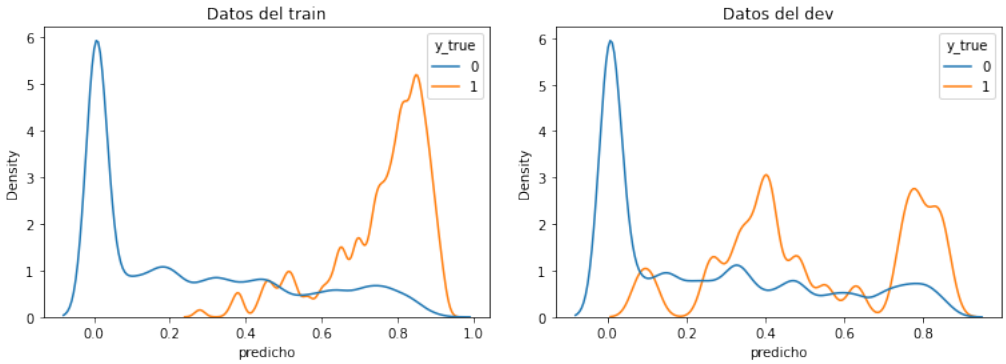

Figura 13: distribución de probabilidades para cada una de las clases utilizando los datos de entrenamiento y validación con el modelo XGBoost.

Por último, cuando optimizamos los hiper parámetros sería deseable que el modelo pueda igualmente separarlos. Esto se comprueba efectivamente en el gráfico de la figura 14, donde se observa la distribución de predicciones en el dev del mejor modelo xgboost.

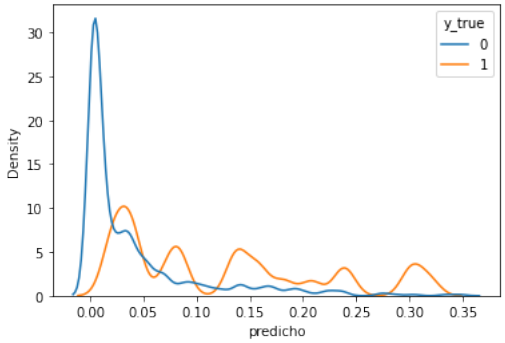

Figura 14: distribución de probabilidades para cada una de las clases utilizando los datos de validación con el modelo XGBoost con hiperparámetros optimizados.

Retomando nuestro objetivo de que el score devuelto por el modelo representase la probabilidad de pertenencia a la clase target correspondiente, realizamos los siguientes gráficos de curva de calibración (cf. figuras 15 y 16). Estas curvas indican cualitativamente cuán descalibrados estaban nuestros modelos, es decir, cuán poco consistentes eran los scores predichos con la probabilidad empírica.

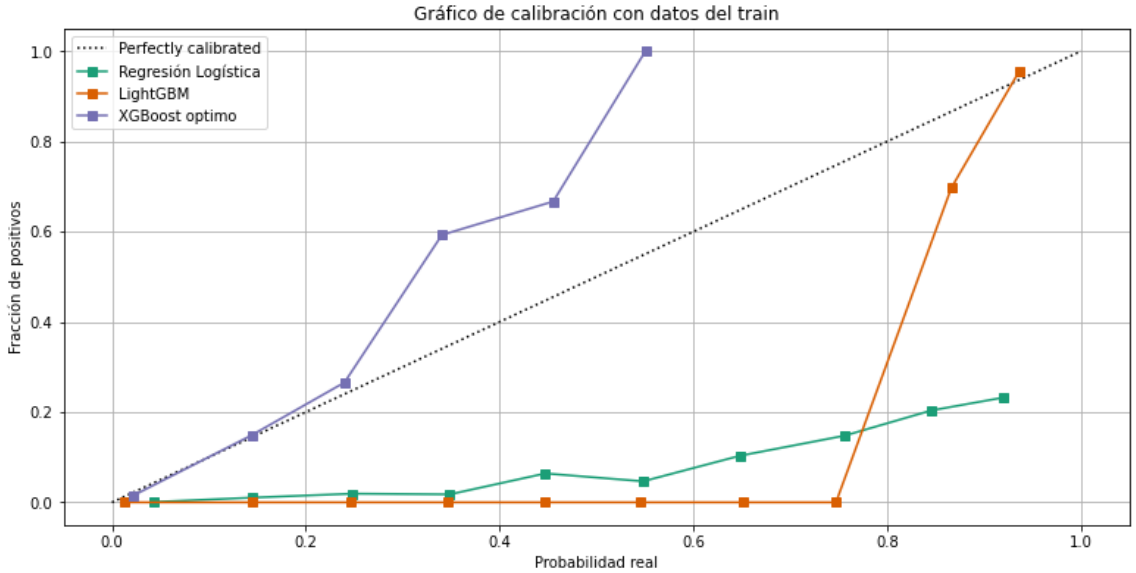

Figura 15: se muestra como evaluando con los mismos datos de entrenamiento aun así las probabilidades estimadas y reales no coinciden, a excepción del final con LGBM ya qué como vimos en la figura 5 entrega scores bruscos y extremos.

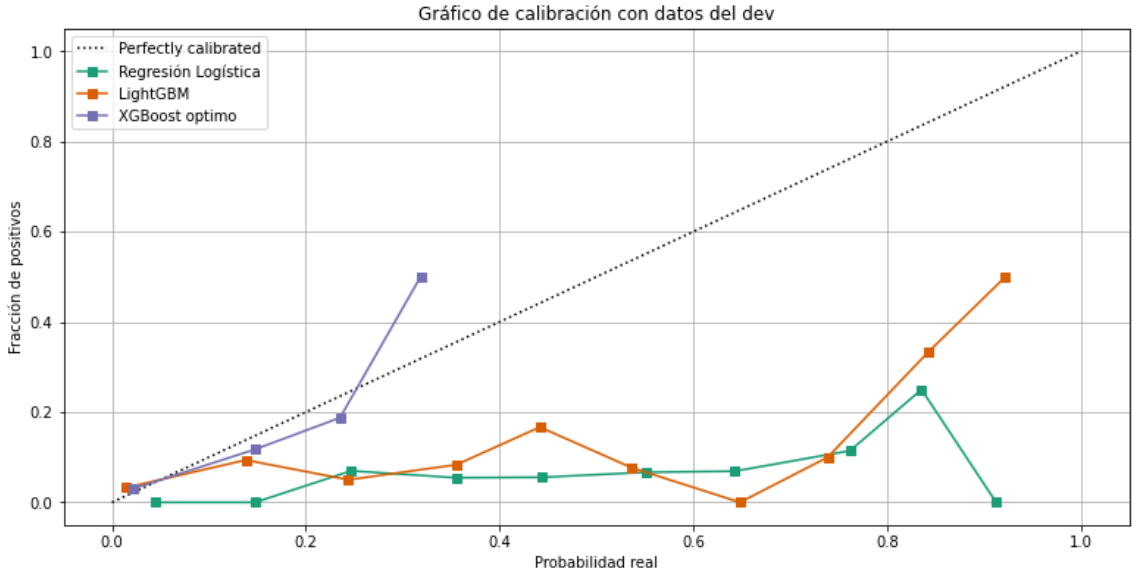

Figura 16: Mismo gráfico que figura 7 pero evaluando en datos de desarrollo que no fueron usados para entrenar. Y claramente sigue teniendo un rendimiento paupérrimo.

Además, recordando que queríamos minimizar la diferencia entre probabilidad estimada y probabilidad real, y por esta razón, tomamos como métrica la log loss (cross entropy), los modelos más coherentes para arreglar este tipo de situaciones son Platt scaling e isotonic regression. Lo que terminamos haciendo fue un pequeño modelo que tomaba de features la score y la score al cuadrado (lo probamos con las scores del logistic regression) y lo entrenamos con la verdaderas y target (cf. figuras 17). Se ve que claramente en el gráfico de calibración las scores de las transformadas se asemejan muchísimo más a las empíricas. Tanto en los datos de entrenamiento (izquierda) como los de desarrollo (derecha).

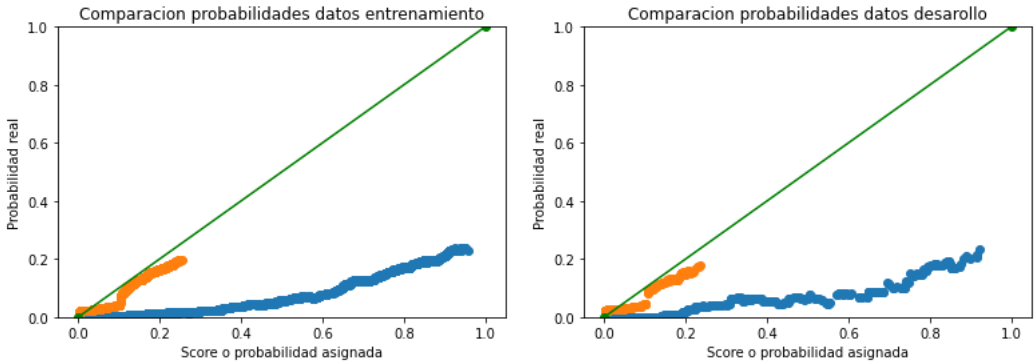

Figura 17: En este gráfico comparamos en azul las scores proveídas por el logistic regression, y por otro lado en naranja las scores transformadas cuadráticamente.

Nótese (cf. figura 18) que la transformación usada es monótona, es decir preserva el orden anterior pero cambia los números.

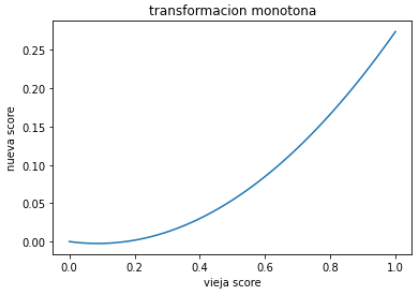

Figura 18: visualización de la transformación. Función que va de la score o probabilidad anterior proveniente del LR y devuelve la nueva score.

Luego de haber transformado las scores obtenemos las siguientes distribuciones de predicciones.

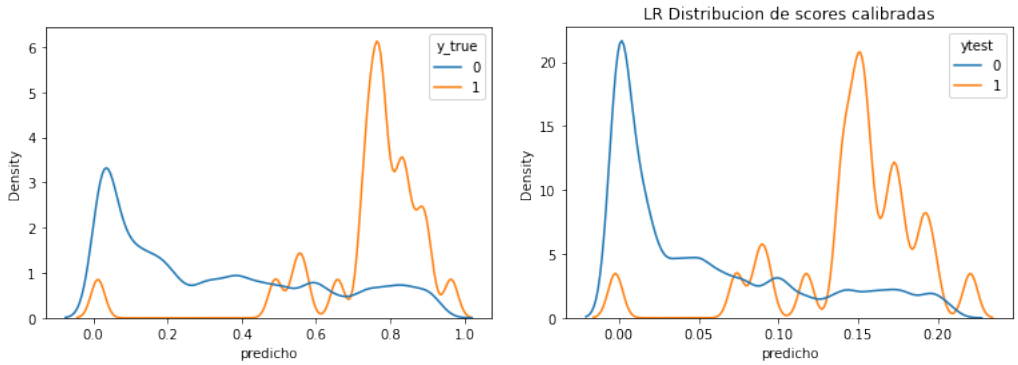

Figura 19: distribuciones de probabilidades antes y luego de aplicarle la calibración de probabilideades usando los datos del test. Tener en cuenta que el eje x del gráfico de la derecha llega hasta aprox. 0.2

Las nuevas scores transformadas ya no se parecen en nada (cf. gráficos de la figura 19), y aunque más modestos ya que ahora intentan inferir probabilidades, podemos observar

que las personas que tuvieron ACV no todas tenían el mismo nivel de riesgo. Hay gente que tuvo un ACV porque tenía muchos factores de riesgo y hay un pequeño grupo de personas que no tenía prácticamente ningún factor. Otra cosa llamativa es que nuestro modelo caracteriza excelentemente a las personas sin riesgo (pega sus probabilidades a 0).

	Sin calibrar	Calibrando
Train	0.500	0.162
Dev	0.468	0.159
Test	0.541	0.211

Tabla 5: Resultados del log loss luego de calibrar logistic regression. Errores de log loss para cada uno de los 3 sets de datos. Teniendo en cuenta que el modelo de regresión logística fue entrenado con train, el modelo calibrador fue entrenado con dev, y que test nunca vió los datos ni fue usado para armar la calibración.

Nos dimos cuenta de que si bien para la regresión logística mejora considerablemente la métrica de log loss después de calibrar (cf. tabla 5), este no era el caso para el XGB hypertuneado (cf. tabla 6). Para el XGB la log loss era ligeramente menor (o sea mejor) sin calibrar en comparación con el LR calibrado, pero al calibrar empeoraba.

	Sin calibrar_XGB	Calibrando_XGB
Train	0.136	0.149
Dev	0.160	0.168
Test	0.150	0.160

Tabla 6: Resultados del log loss para los tres set de datos antes y despues de calibrar

2. Modelo final y conclusiones

Como modelo final elegimos el XGBoost con los hiperparámetros optimizados sin calibración de probabilidades, esto es teniendo en cuenta la menor log loss tanto en dev (0.16) como en test (0.15) en relación a los otros modelos probados (cf. tabla 7). En el proceso tuvimos que entender el porqué de los scores que devolvía el modelo y darnos cuenta de que diferenciar las dos clases no es suficiente, también hay que poder cuantificar las probabilidades de tener ACV dado que ya de por sí es algo muy raro. En conclusión, este sería el modelo final que le presentaremos a un equipo de especialistas para luego desarrollar la solución final. Esto se debe a que el error es bajo, tanto en dev como en el test, mostrando robustez frente a los otros modelos comparados. Asimismo, destacamos que el score dado por el modelo se aproxima a la probabilidad de que un individuo tenga ACV, facilitando la interpretación del resultado del modelo.

name	train	dev	test
lr calibrado	0.162	0.159	0.211
solo lgb	0.112	0.250	0.248
solo xgb	0.374	0.401	0.441
xgb + hyperopt	0.136	0.160	0.150

Tabla 7: comparación de los modelos finales