Inteligencia de Negocio - Práctica 2
Análisis Relacional mediante Segmentación

Sergio Quijano Rey - 72103503k
sergioquijano@correo.ugr.es
5º Doble Grado Ingeniería Informática y Matemáticas
Grupo de prácticas 1

31 de diciembre de 2021

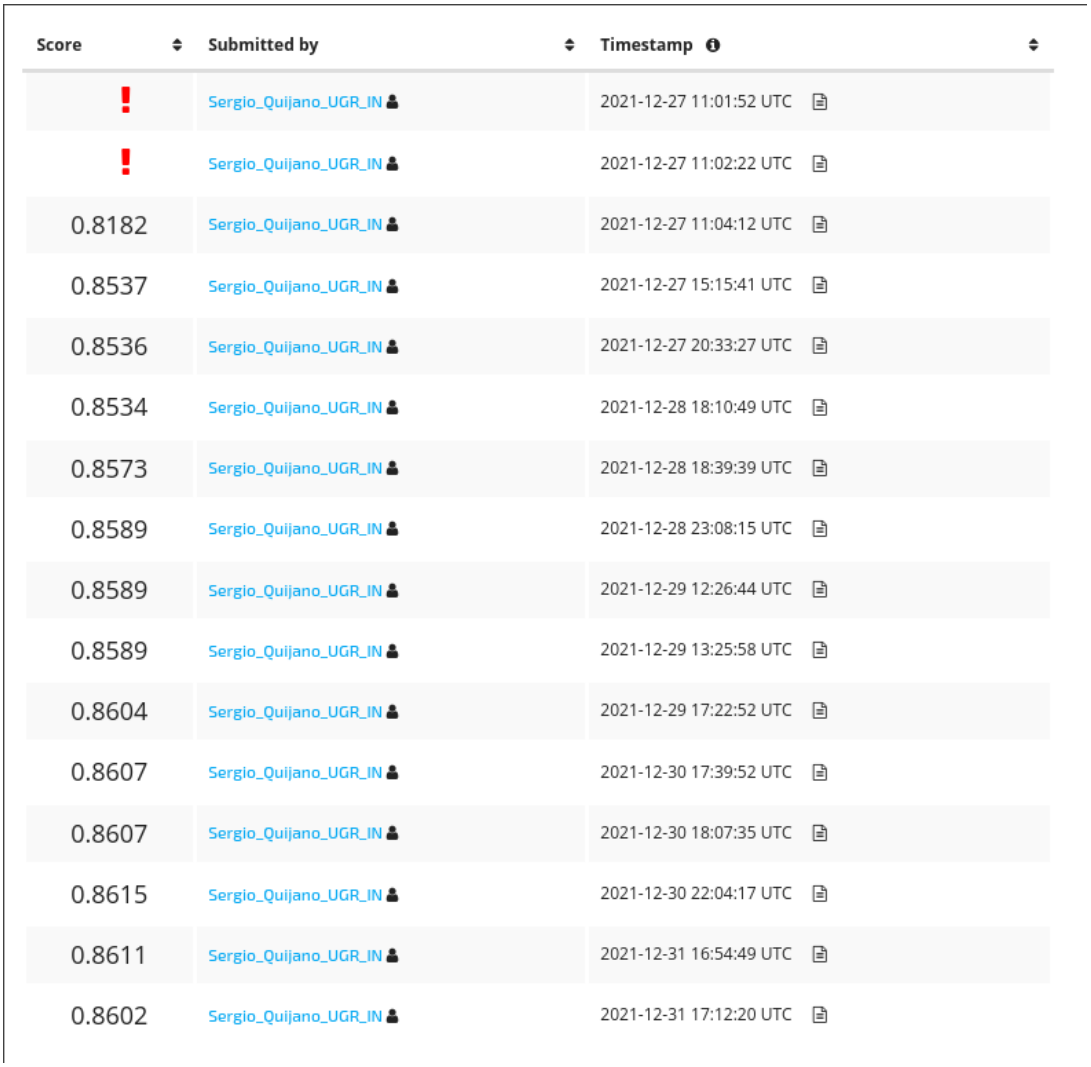
Índice

Índice de figuras

Índice de cuadros

1. Tabla submissions

A continuación mostramos la tabla de *submissions* en la que queda reflejado el esfuerzo por mejorar los resultados a lo largo de los días que nos hemos centrado en desarrollar el código para la práctica:



Score	Submitted by	Timestamp
!	Sergio_Quijano_UGR_IN	2021-12-27 11:01:52 UTC
!	Sergio_Quijano_UGR_IN	2021-12-27 11:02:22 UTC
0.8182	Sergio_Quijano_UGR_IN	2021-12-27 11:04:12 UTC
0.8537	Sergio_Quijano_UGR_IN	2021-12-27 15:15:41 UTC
0.8536	Sergio_Quijano_UGR_IN	2021-12-27 20:33:27 UTC
0.8534	Sergio_Quijano_UGR_IN	2021-12-28 18:10:49 UTC
0.8573	Sergio_Quijano_UGR_IN	2021-12-28 18:39:39 UTC
0.8589	Sergio_Quijano_UGR_IN	2021-12-28 23:08:15 UTC
0.8589	Sergio_Quijano_UGR_IN	2021-12-29 12:26:44 UTC
0.8589	Sergio_Quijano_UGR_IN	2021-12-29 13:25:58 UTC
0.8604	Sergio_Quijano_UGR_IN	2021-12-29 17:22:52 UTC
0.8607	Sergio_Quijano_UGR_IN	2021-12-30 17:39:52 UTC
0.8607	Sergio_Quijano_UGR_IN	2021-12-30 18:07:35 UTC
0.8615	Sergio_Quijano_UGR_IN	2021-12-30 22:04:17 UTC
0.8611	Sergio_Quijano_UGR_IN	2021-12-31 16:54:49 UTC
0.8602	Sergio_Quijano_UGR_IN	2021-12-31 17:12:20 UTC

Figura 1: Tabla de *Driven Data* con la información de todas las propuestas realizadas

En la figura anterior podemos ver que las dos primeras entregas no tuvieron *score*. Esto se debe a que subimos los resultados con un formato erróneo.

2. Introducción

2.1. Descripción del problema

Según la descripción del problema de *Driven Data* [1], debemos calcular la probabilidad de que una persona se vacune de dos tipos de vacunas distintas. Es decir, deberemos calcular dos probabilidades, una por cada tipo de vacuna con la que se trabaja. Las vacunas son para el virus *h1n1* y para la gripe.

Disponemos de 39 columnas. Una de ellas es para el identificador de la persona encuestada (que no usaremos) y dos de ellas son las etiquetas a predecir. Por tanto disponemos de 36 columnas para llevar a cabo la tarea de aprendizaje. Tenemos columnas de tipo numérico y de tipo categórico. Por otro lado, tenemos 26707 ejemplos en nuestra base de datos de entrenamiento.

La métrica a optimizar, y en la que se basará el *ranking* de la competición, será el área bajo la curva ROC. Como tenemos dos etiquetas a predecir, será la media del área bajo la curva ROC para las dos etiquetas por separado.

Por tanto, las propuestas subidas a la plataforma *Driven Data* deberán ser dos valores probabilísticos entre 0 y 1, y no simplemente valores de clasificación binaria $\{0, 1\}$.

Respecto a la plataforma donde se desarrolla la competición, dejan a nuestra disposición dos conjuntos de datos. El conjunto de entrenamiento, etiquetado, y el conjunto de test, sin etiquetar. A partir del conjunto de test deberemos generar la propuesta que subimos a la plataforma. Además, solo dispondremos de 3 propuestas al día. Cuando se realiza una propuesta, se conoce el *score* en test de forma inmediata.

2.2. Consideraciones iniciales

En primer lugar, en el fichero que subimos a *PRADO*, el código se encuentra separado en carpetas enumeradas, una carpeta por cada experimento realizado. En cada carpeta se encuentra el *Notebook* conteniendo todo el código empleado, el fichero `final_submission.csv` con la propuesta realizada, y dos capturas de pantalla. Una captura con el *score* obtenido en la propuesta, y otra captura que muestra la posición ocupada en el momento de realizar la propuesta.

En segundo lugar, todos los *Notebooks* tendrán una sección inicial con funciones comunes a todo el código. Por ejemplo, funciones para evaluar modelos, para trabajar con *dataframes*, ...

En tercer lugar, también tendremos en todos los *Notebooks* una variable `RUNNING_ENV`. Cuando es `"local"`, indicamos que estamos corriendo el código en nuestra máquina. Cuando es `"remote"`, indicamos que estamos corriendo en *Google Colab*. Con esto controlamos diferencias sutiles, como la autorización necesaria en *Google Colab*, o las rutas a la carpeta de datos. Y con ello, evitamos tener que manejar dos bases de código según el entorno en el que estemos corriendo.

Hemos empleado la herramienta *Google Colab* intensamente. Principalmente a la hora de realizar *Hyperparameter Tuning* usando *Cross Validation* para ello. Casi todos los algoritmos han tardado aproximadamente 90 o 120 minutos en terminar su *tuning*, y con la cantidad de algoritmos considerados, hemos llegado a tardar 14h en realizar este proceso de principio a fin.

Por tanto, gracias al uso de *Google Colab*, hemos podido dejar esta búsqueda en un segundo plano, para seguir mejorando el código en local, con nuestra máquina sin sobrecargar por este proceso exigente computacionalmente.

En cuarto lugar, subimos también a *PRADO* el archivo `pyproject.toml`, en el que definimos todos los paquetes usados para el desarrollo de la práctica. Algunos paquetes pueden no encontrarse en las instalaciones por defecto de entornos de *Data Science* habituales (como pudiera ser `conda`). Por tanto, y para permitir reproducir los experimentos independientemente del entorno, especificamos los paquetes empleados en este fichero (que se corresponde al gestor de paquetes `poetry`).

En quinto lugar, se podrá consultar el código de esta práctica (y de todas las prácticas de la asignatura) en el repositorio de *Github* `https://github.com/SergioQuijanoRey/PracticasInteligenciaNegocio`. Este repositorio será público a partir del 7 de enero, momento en el que las prácticas de la asignatura finalizan.

En sexto y último lugar, por falta de tiempo (pues hemos tenido que preparar otras asignaturas), solo hemos realizado propuestas hasta el 31 de Diciembre de 2021. Tenemos más ideas de cómo continuar mejorando los resultados, pero no disponemos del tiempo para ello. Esto se deja de manifiesto en *"4. Conclusiones"*, donde se comenta la posición en la competición en esa fecha.

3. Desarrollo de los experimentos

3.1. Resumen de los experimentos realizados

En la siguiente tabla mostramos un resumen del desarrollo de los experimentos. El desarrollo de los experimentos se ha realizado de forma incremental. Es decir, hemos partido del experimento anterior, y hemos añadido mejoras o cambios, en la mayoría de los casos no disruptivos. En caso de un cambio de gran calado se indicará en la siguiente tabla. Por lo tanto, la siguiente tabla recoge los cambios de forma incremental.

Por la longitud de las tabla global, separaremos dicha tabla en dos tablas. Una tabla que muestre los resultados obtenidos en la competición, y otra tabla que explique los cambios realizados. Mostramos ambas tablas a continuación:

Propuesta	Fecha y hora (hora española)	Posición ocupada	Score Training	Score Test Driven Data
1	27/12/2021 12:08	793	-	0.8182
2	27/12/2021 16:17	398	0.8658	0.8537
3	27/12/2021 21:33	402	0.8426	0.8536
4	28/12/2021 19:12	405	0.85966	0.8534
5	28/12/2021 19:40	339	0.8630	0.8573
6	29/12/2021 00:09	305	0.8823	0.8589
7	29/12/2021 13:27	305	0.8646	0.8589
8	29/12/2021 14:26	305	0.8648	0.8589
9	29/12/2021 18:25	248	0.8641	0.8604
10	30/12/2021 18:43	244	0.8644	0.8607
11	30/12/2021 19:08	244	0.86452	0.8607
12	30/12/2021 23:05	199	0.8650	0.8615
13	31/12/2021 17:55	202	0.8643	0.8611
14	31/12/2021 18:17	202	0.8642	0.8602

Cuadro 1: Resumen de los resultados logrados con los cambios incrementales a lo largo de la práctica

Nombre Propuesta	Descripción preprocesado	Descripción Algoritmo	Configuración de parámetros
1	Nos quedamos solo con las variables numéricas, imputamos los missing values usando la mediana	Regresión logística, entrenando dos modelos para las dos variables objetivo	C = 1, regularización l2
2	Imputamos missing values con la mediana. Normalizamos a media 0 y desviación 1. Nos quedamos con las variables categóricas, que transformamos a numéricas usando *one hot encoding*	Añadimos cross validation. Decidimos usar AdaBoost. Añadimos la evaluación del modelo sobre el conjunto de validación	lr = 0.5, n_estimators = 200
3	Imputamos missing values con un estimador en base a las otras variables. Usamos Smote+TomekLinks. Borramos *outliers* con *Local Outlier Factor*	Volvemos a hacer CV y tomamos los mejores parámetros	lr = 0.75 n_estimatos = 200
4	Borramos SMOTE + TomekLinks, cacheamos el pre-procesado de datos para poder hacer iteraciones más rápidas	Añadimos CatBoost y hacemos CV para seleccionar los mejores hiperparámetros	lr = 0.5, iterations = 20, depth = 4
5	Entrenamos sobre todo el conjunto de datos tras entrenar y evaluar en validación	Catboost al que hemos cambiado manualmente los parámetros	lr = 0.5, iterations = 40, depth = 4
6	-	Ajustamos CV para todos los modelos, incluido Random Forest. Sigue siendo el mejor Catboost	lr = 0.25, iterations = 80, depth = 4
7	-	Ensemble con los cuatro algoritmos, con los mejores parámetros que hemos encontrado	Logistic: penalty = "l2", C = 0.05 ; Adaboost: n_estimators = 200, learning_rate = 0.5 ; Catboost: iterations=80, learning_rate=0.25, depth=4 ; Random Forest: n_estimators = 200, criterion = ."entropy", min_samples_split = 4, min_samples_leaf = 3

Cuadro 2: Resumen de los cambios incrementales realizados a lo largo de la práctica, primera parte

Nombre Propuesta	Descripción preprocesado	Descripción Algoritmo	Configuración de parámetros
8	-	Ponderamos el ensemble con los resultados de los modelos individuales, en validación, aplicando softmax	-
9	Además del borrado de outliers que teníamos, añado IsolationForest para borrar outliers	Aplicamos correctamente el ensemble (estábamos re-entrenando con un Catboost simple)	-
10	En vez de usar imputación con estimador, imputamos usando la mediana	Añadimos Extreme Random Forest y MLP. Hacemos hyperparamter tuning de todos los algoritmos que llevamos explorados hasta ahora. Añadimos los dos nuevos modelos al ensemble	Mismos parámetros para modelos ya explorados. Extreme random forest: nestimators: 300, min_samples_split: 2, min_samples_leaf: 3. MLP: alpha: 0.1, learning_rate_init: 0.0001
11	En la normalización de datos, aprendemos los parámetros usando training, validación y test	-	-
12	Deshago la normalización de datos usando todos los conjuntos. Solo uso el conjunto de entrenamiento para aprender los parámetros	Añado XGBOOST. Exploro sus parámetros con CV. Lo añado al ensemble de modelos	XGBOOST: max_depth: 4, eta: 0.15
13	-	Uso ranking en vez de softmax para realizar la asignación de pesos	-
14	-	Uso solo MLP, Catboost y Xgboost para el ensemble. Volvemos a usar softmax para los pesos del ensemble, en vez del ranking	-

Cuadro 3: Resumen de los cambios incrementales realizados a lo largo de la práctica, segunda parte

Notar que en *"1. Resumen de los resultados logrados con los cambios incrementales a lo largo de la práctica"*, en la primera propuesta, no tenemos resultados de *training*. Esto se debe a que, como comentaremos más adelante, en esta primera propuesta todavía no habíamos desarrollado un método de evaluación de los resultados, y por tanto no tuvimos disponible esa métrica. Y notar también que, en *"**??**. **??**"*, cuando no hacemos cambios o bien en los algoritmos o bien en el tratamiento de los datos, indicamos con un *"–"* dicha ausencia de cambios, evitando así ser excesivamente repetitivos.

Con todo esto, pasamos a desarrollar, propuesta a propuesta, el trabajo desarrollado.

3.2. Experimento 1

En este primer experimento, escribimos el mínimo código para poder hacer una *submission* y familiarizarnos así con la plataforma. Por tanto, y como se puede apreciar en *"1. Resumen de los resultados logrados con los cambios incrementales a lo largo de la práctica"*, no tenemos información sobre el rendimiento de nuestro modelo en *training*.

Parte de nuestro código lo basamos en el código que *Driven Data* muestra como *benchmark* [2], aunque gran parte del código la tomamos de una plantilla nuestra, preparada con las tareas más repetitivas de todo proyecto de *Data Science*. En concreto, tomamos el modelo propuesto en el ejemplo, y el código para realizar la propuesta.

El pre-procesado de los datos es muy básico:

1. Empezamos borrando la columna `"respondent_id"`. No nos hace falta, pues el `dataframe` ya guarda en su estado interno un identificador para las filas
2. Filtramos todas las variables categóricas, solo nos quedamos con las numéricas
3. Separamos el conjunto de entrenamiento en entrenamiento (80 %) y validación (20 %)
4. Imputamos los *missing values* usando la mediana

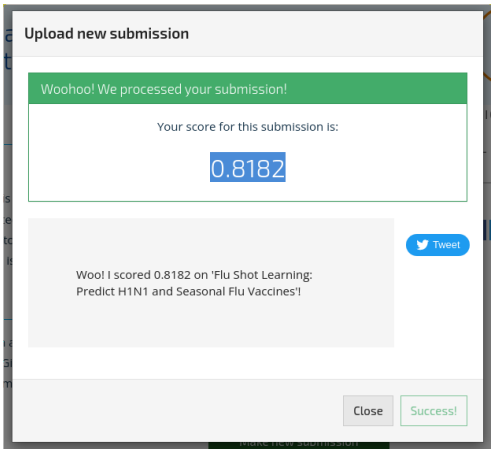
En este pre-procesado, y en todos los siguientes, cuando realizamos una transformación sobre el conjunto de entrenamiento con parámetros que se aprenden, se aplicará en validación y test con dichos parámetros, aprendidos en entrenamiento. Si aprendiésemos dichos parámetros sobre todo el conjunto de datos (*training*, validación y test juntos) estaríamos cometiendo *data snooping*.

A la hora de separar en *training* y *validation*, usamos estratificación por la etiqueta de salida. Como tenemos dos etiquetas, las combinamos en una sola, estratificamos usando esa etiqueta combinada, y deshacemos el combinado de etiquetas.

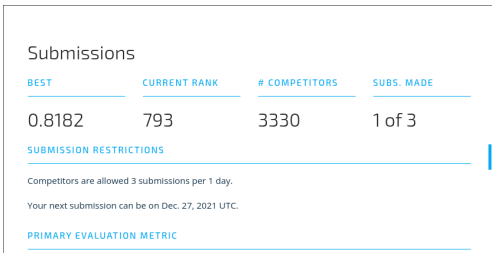
El modelo aplicado es muy básico: dos modelos de regresión logística. En este caso, y en todos los posteriores, entrenaremos dos modelos del mismo tipo para clasificar las dos etiquetas. Esto lo haremos cómodamente con la clase `MultiOutputClassifier`. De ahora en adelante, este detalle técnico lo omitiremos y hablaremos directamente de modelos, entendiendo por ello lo que aquí detallamos.

Es en este experimento en el que realizamos dos propuestas fallidas, como muestra *"1. Resumen de los resultados logrados con los cambios incrementales a lo largo de la práctica"*.

Los resultados se muestran en las siguientes figura:



(a) Score obtenida en la *submission*



(b) Posición de ese momento en la competición

Figura 2: Resultados obtenidos tras hacer la *submission* en la plataforma *online*

3.3. Experimento 2

El pre-procesado en este experimento es:

1. Seguimos imputando los *missing values* con la mediana
2. Normalizamos el conjunto de datos a media 0 y desviación típica 1
3. En vez de eliminar las variables numéricas, las pasamos a numéricas con *one hot enconding* y las consideramos en nuestro conjunto de entrenamiento

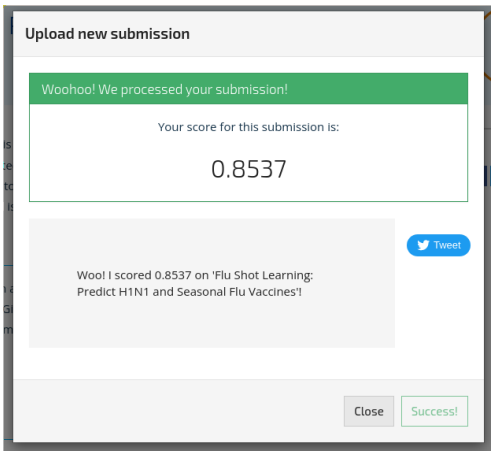
Además de ese pre-procesado de los datos, añadimos el modelo *Adaboost*. Escribimos un *GridSearch* propio para hacer *Hyper Parameter Tuning*. Con esto nos referimos a que especificamos los conjuntos de parámetros manualmente, así como el bucle en que evaluamos con *10-fold Cross Validation*. Con esto tenemos un mayor control en este proceso que puede ser muy largo. Por ejemplo, podemos parar la búsqueda en la mitad del proceso, y tomar el mejor parámetro encontrado hasta el momento. O definir los *logs* que queremos mostrar durante el proceso. Esto no lo podíamos hacer con la funcionalidad correspondientes de `sklearn`, y al no suponer mucho código, lo implementamos nosotros para resolver nuestras necesidades.

Aplicamos dicha búsqueda a los dos modelos que hemos tratado hasta ahora. Empleamos *Adaboost* por dar mejores resultados.

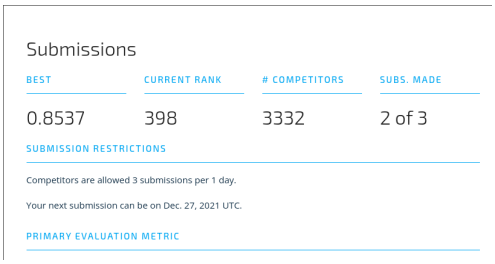
Además, añadimos la evaluación del modelo. Para ello, llamamos a la función propia `evaluate_model` , en la que calculamos algunas métricas adicionales.

Todavía no hemos implementado el re-entrenamiento sobre todo el conjunto de datos, así que la *submission* se realiza con el modelo entrenado en un subconjunto de los datos iniciales.

Los resultados se muestran en las siguiente figura:



(a) Score obtenida en la *submission*



(b) Posición de ese momento en la competición

Figura 3: Resultados obtenidos tras hacer la *submission* en la plataforma *online*

3.4. Experimento 3

Los cambios en el pre-procesado de datos son:

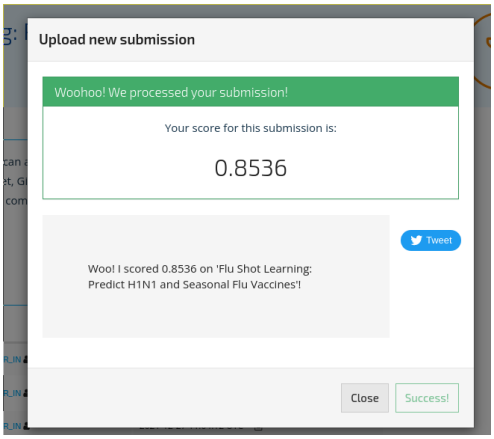
- En vez de imputar *missing values* usando la mediana, usamos predictores en el resto de variables
- Para balancear los datos (la etiqueta `h1n1` tiene mucho desbalanceo) usamos *smote + tomek links*
- Para el borrado de *outliers*, usamos una técnica avanzada como es *Local Outlier Factor*

De nuevo, para hacer el balanceo con *smote + tomek links*, combinamos las dos variables de salida en una única, realizamos el balanceo, y descombinamos dicha etiqueta única.

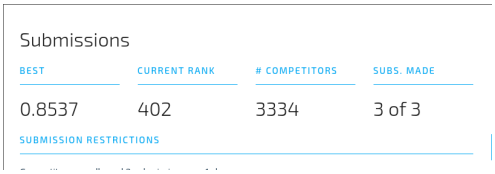
Respecto a los algoritmos, esta modificación al conjunto de datos (el balanceo y el borrado de *outliers*) nos obliga a repetir *Cross Validation* para tomar los mejores parámetros, del conjunto de parámetros que exploramos. Seleccionamos de nuevo *AdaBoost* por ser el mejor modelo de los dos que estamos considerando.

Como se ve en *"1. Resumen de los resultados logrados con los cambios incrementales a lo largo de la práctica"*, este pre-procesado de datos no consigue mejorar los resultados, es más, los empeora ligeramente.

Los resultados se muestran en las siguiente figura:



(a) Score obtenida en la *submission*



(b) Posición de ese momento en la competición

Figura 4: Resultados obtenidos tras hacer la *submission* en la plataforma *online*

3.5. Experimento 4

Los cambios en el pre-procesado de datos son:

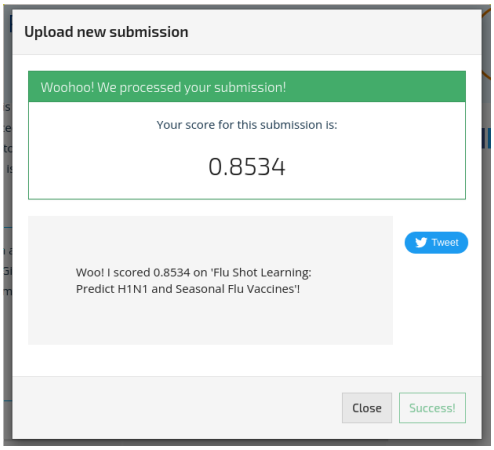
1. Borramos el balanceo de los datos usando *Smote + Tomek Links* por los malos resultados que producen
2. Añadimos una variable global que nos permite cachear el procesado de datos

Cacheando el procesado de datos, podemos evitar realizar este cómputo de forma repetitiva en caso de que no hayamos cambiado nada en iteraciones previas (en otro experimento o en el mismo experimento). Si cacheamos, nos saltamos el procesado de datos y cargamos los datos desde la caché (un fichero de `numpy`). En otro caso, realizamos el pre-procesado y actualizamos la caché.

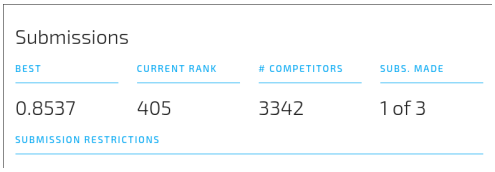
Respecto a los cambios en los algoritmos:

1. Añadimos *Catboost* a la lista de algoritmos a explorar
2. Exploramos sus parámetros usando *Cross Validation*
3. Seleccionamos *Catboost* como mejor modelo

Los resultados se muestran en las siguiente figura:



(a) Score obtenida en la *submission*



(b) Posición de ese momento en la competición

Figura 5: Resultados obtenidos tras hacer la *submission* en la plataforma *online*

Seguimos sin mejorar los resultados.

3.6. Experimento 5

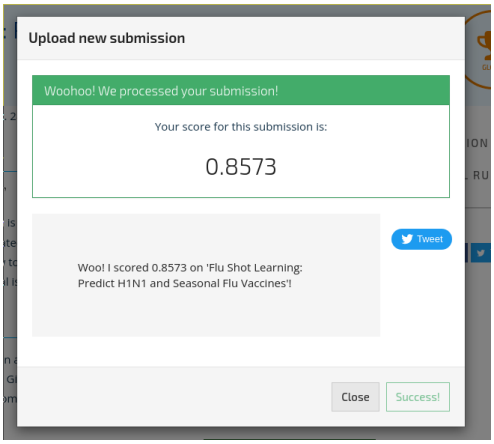
Respecto al tratamiento de los datos:

1. En este experimento, después de entrenar sobre el conjunto de entrenamiento y evaluar sobre el conjunto de evaluación, entrenamos sobre todo el conjunto de datos. Esto lo hacemos juntando de nuevo entrenamiento y validación

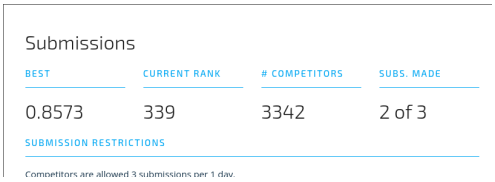
Respecto al uso de modelos:

1. Ajustamos los parámetros de *Catboost* de forma manual
2. Con esto nos damos cuenta de que no estamos explorando de forma correcta el espacio de parámetros de este algoritmo

Los resultados se muestran en las siguiente figura:



(a) Score obtenida en la *submission*



(b) Posición de ese momento en la competición

Figura 6: Resultados obtenidos tras hacer la *submission* en la plataforma *online*

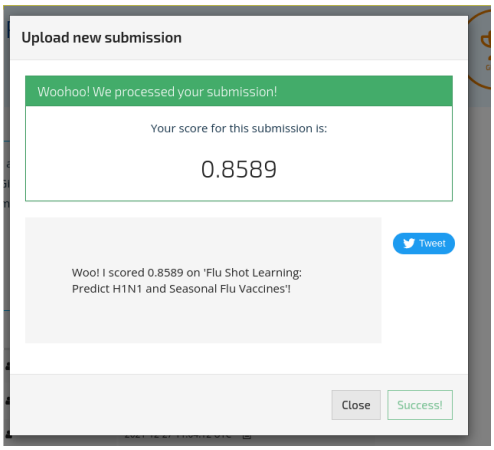
En este caso sí que vemos una mejora notable en los resultados de la competición. Con esto, en el siguiente experimento, realizaremos una exploración de los parámetros de *Catboost* mucho más concienzudo.

3.7. Experimento 6

Mantenemos igual el pre-procesado de los datos. Respecto al uso de modelos:

1. Exploramos los parámetros de los modelos, usando *Cross Validation*, de forma más concienzuda.
2. Anteriormente los parámetros óptimos se alcanzaban en las fronteras, lo que indica que hay que seguir expandiendo dicha frontera para encontrar aún mejores parámetros
3. Incluimos Random Forest a nuestro conjunto de modelos a explorar
4. Exploramos sus parámetros usando *Cross Validation*
5. Seguimos usando *Catboost* por sus buenos resultados
6. Por la cantidad de modelos que estamos explorando, añadimos un diccionario con los mejores resultados de cada modelo, que mostramos conjuntamente tras realizar todos los procesos de *Cross Validation*

Los resultados se muestran en la siguiente figura:



(a) Score obtenida en la *submission*

(b) Posición de ese momento en la competición

Figura 7: Resultados obtenidos tras hacer la *submission* en la plataforma *online*

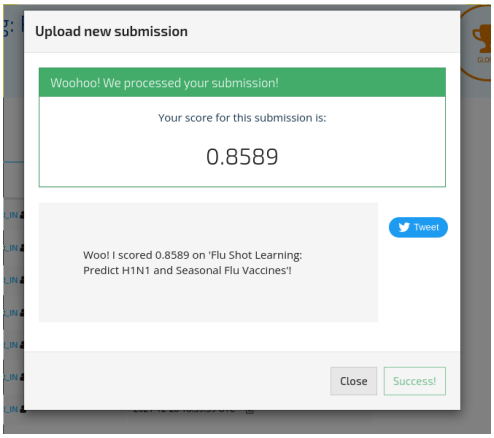
Conseguimos mejorar progresivamente los resultados en la plataforma *Driven Data*.

3.8. Experimento 7

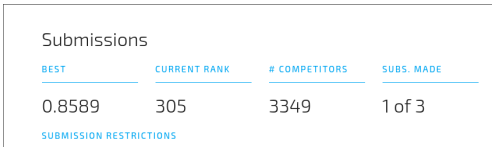
Mantenemos igual el pre-procesado de los datos. Respecto al uso de modelos:

1. Realizamos un **cambio fundamentan en el enfoque** usado en el resto de la práctica
2. Construimos un *ensemble* con los cuatro modelos con los que estamos trabajando. Usamos *soft voting* porque necesitamos realizar predicciones probabilísticas y no predicciones binarias (que correspondería a voto mayoritario)
3. Usamos los mejores parámetros encontrados usando *Cross Validation*
4. Por un error humano, re-entrenamos sobre todo el conjunto de datos usando *Catboost* y no todo el *ensemble*. Por tanto, no será visible el cambio producido por este cambio de paradigma

Los resultados se muestran en la siguiente figura:



(a) Score obtenida en la *submission*



(b) Posición de ese momento en la competición

Figura 8: Resultados obtenidos tras hacer la *submission* en la plataforma *online*

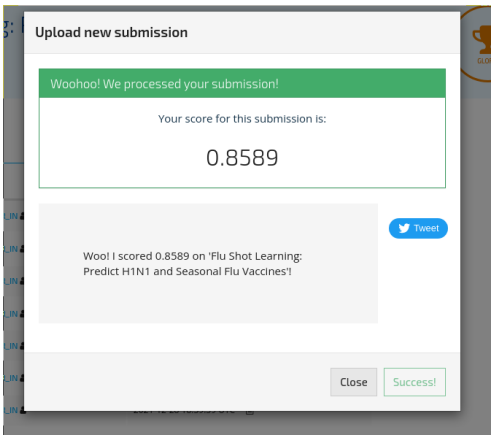
Vemos que no mejoramos por el error que ya hemos comentado.

3.9. Experimento 8

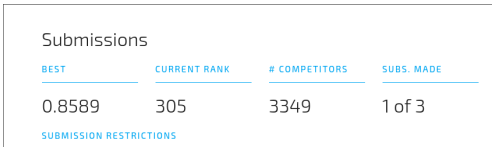
Mantenemos igual el pre-procesado de los datos. Respecto al uso de modelos:

1. Ponderamos la importancia de cada modelo del *ensemble* usando los resultados de los modelos, de forma individual, sobre el conjunto de validación
2. Para ello usamos los *scores* sobre validación (el área bajo la curva ROC) y aplicamos *softmax* para normalizar estos resultados y usarlos como vector de pesos
3. Por el mismo error , re-entrenamos sobre todo el conjunto de datos usando *Catboost* y no todo el *ensemble*. Por tanto, no será visible el cambio producido por este cambio de paradigma

Los resultados se muestran en la siguiente figura:



(a) Score obtenida en la *submission*



(b) Posición de ese momento en la competición

Figura 9: Resultados obtenidos tras hacer la *submission* en la plataforma *online*

Vemos que no mejoramos por el error que ya hemos comentado.

3.10. Experimento 9

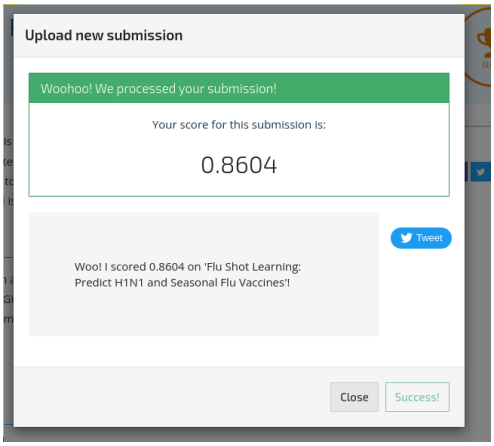
Respecto al procesado o manejo de los datos:

1. Al borrado de *outliers* usando *Local Outlier Factor*, añadimos (sin sustituir) borrado de *outliers* usando *Isolation Forest*

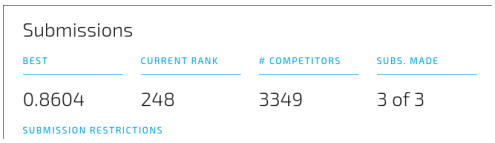
Respecto al uso de modelos:

1. Corregimos el error que hemos tenido en los dos modelos previos
2. Al re-entrenar sobre todo el conjunto de datos, usamos el *ensemble* construido y ponderado previamente

Los resultados se muestran en la siguiente figura:



(a) Score obtenida en la *submission*



(b) Posición de ese momento en la competición

Figura 10: Resultados obtenidos tras hacer la *submission* en la plataforma *online*

En este caso sí que podemos ver una mejora significativa respecto a los experimentos anteriores. Por tanto, seguiremos durante unos cuantos experimentos expandiendo la potencia del *ensemble*.

3.11. Experimento 10

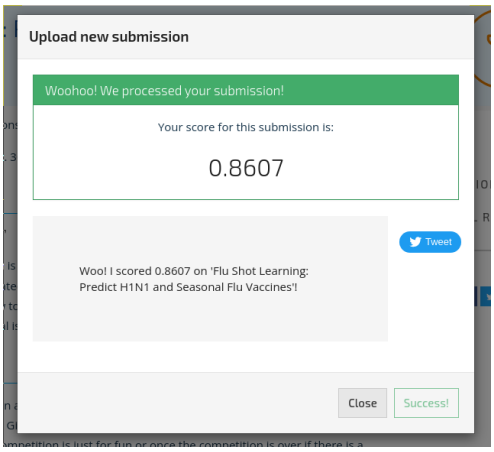
Respecto al procesado o manejo de los datos:

1. En vez de imputar *missing values* con un estimador sobre el resto de variables, volvemos a imputar con la mediana, que dio muy buenos resultados en los primeros experimentos

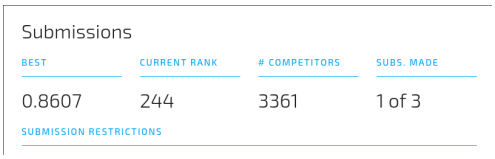
Respecto al uso de modelos:

1. Añadimos los modelos *Extreme Random Forest* y *MLP*
2. Exploramos sus parámetros con *Cross Validation*
3. Al cambiar algo el conjunto de datos, exploramos los parámetros de todos los algoritmos que llevamos estudiados hasta ahora
4. Añadimos estos dos nuevos modelos, con sus mejores parámetros, al *ensemble*

Los resultados se muestran en la siguiente figura:



(a) Score obtenida en la *submission*



(b) Posición de ese momento en la competición

Figura 11: Resultados obtenidos tras hacer la *submission* en la plataforma *online*

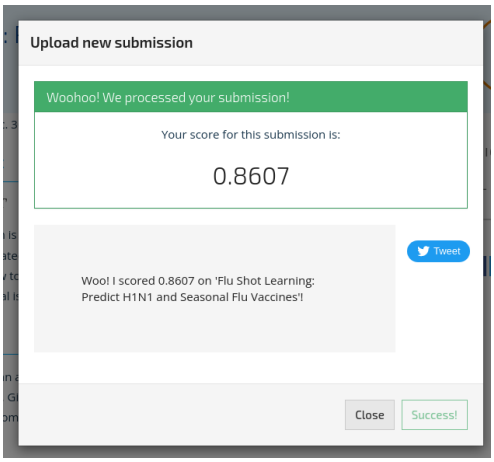
Seguimos mejorando progresivamente los resultados.

3.12. Experimento 11

Respecto al procesado o manejo de los datos:

1. Hasta ahora, hemos normalizado los datos evitando hacer *data snooping*
2. Es decir, aprendemos los parámetros de la normalización (μ, σ) en entrenamiento, y aplicamos esta normalización con estos parámetros en entrenamiento, validación y test
3. Probamos a normalizar considerando, en el aprendizaje de los parámetros, los tres conjuntos de datos
4. Por tanto, estamos cometiendo, de forma consciente, *data snooping*
5. Esto está totalmente desaconsejado en aplicaciones reales en las que el rigor es muy importante para asegurar el buen funcionamiento fuera de datos nunca vistos. Pero aquí esto no es central, sino que lo importante es mejorar los resultados usando todos los datos que la plataforma nos proporciona

Respecto al uso de modelos, no realizamos modificaciones. Los resultados se muestran en la siguiente figura:



(a) Score obtenida en la *submission*

(b) Posición de ese momento en la competición

Figura 12: Resultados obtenidos tras hacer la *submission* en la plataforma *online*

Los resultados no han cambiado. Lo más probable es que la plataforma esté preparada para minimizar el efecto de usar este mal procedimiento. O también es posible que, al estar obteniendo resultados cercanos a los mejores de la competición (en el momento de realizar esta propuesta), este mal procedimiento no tenga un impacto significativo.

3.13. Experimento 12

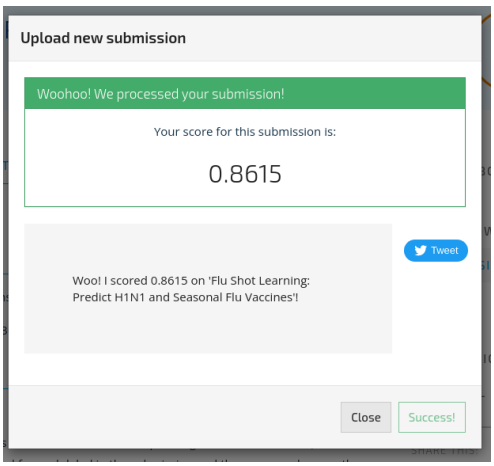
Respecto al procesado o manejo de los datos:

1. Volvemos a realizar una normalización rigurosa, aprendiendo los parámetros en entrenamiento y aplicando dicha transformación en entrenamiento, validación y test

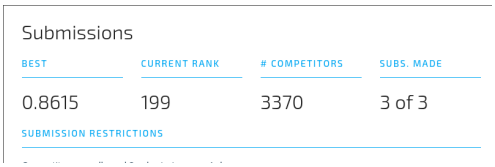
Respecto al uso de modelos:

1. Añadimos *Xgboost* como modelo a estudiar
2. Exploramos sus parámetros usando *Cross Validation*
3. Lo añadimos al *ensemble*

Los resultados se muestran en la siguiente figura:



(a) Score obtenida en la *submission*



(b) Posición de ese momento en la competición

Figura 13: Resultados obtenidos tras hacer la *submission* en la plataforma *online*

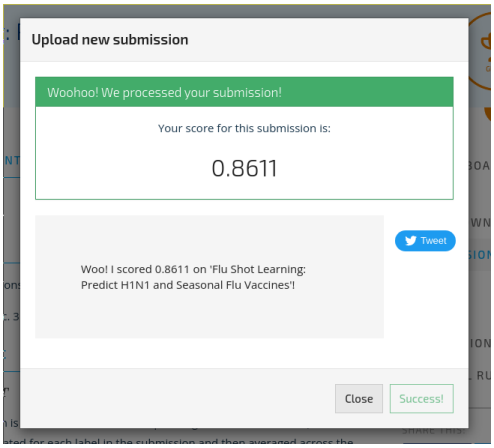
En este caso, vemos una gran mejora al añadir este modelo a nuestro *ensemble*. Además, es la mejor posición que ocupamos en el momento de terminar de realizar experimentos (31/12/2021). Teniendo en cuenta que terminamos en la posición 199 de 3370, esto nos deja en el top 6 % de la competición en ese momento.

3.14. Experimento 13

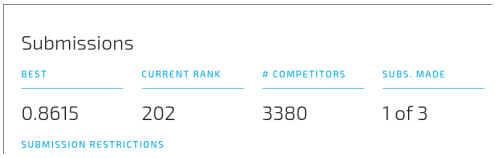
Respecto al procesado o manejo de los datos, no realizamos modificaciones. Respecto al uso de modelos:

1. En vez de aplicar *softmax* sobre los scores de los modelos para generar los pesos del modelo, usamos un *ranking* inverso como pesos (7 para el mejor modelo, 1 para el peor modelo)

Los resultados se muestran en la siguiente figura:



(a) Score obtenida en la *submission*



(b) Posición de ese momento en la competición

Figura 14: Resultados obtenidos tras hacer la *submission* en la plataforma *online*

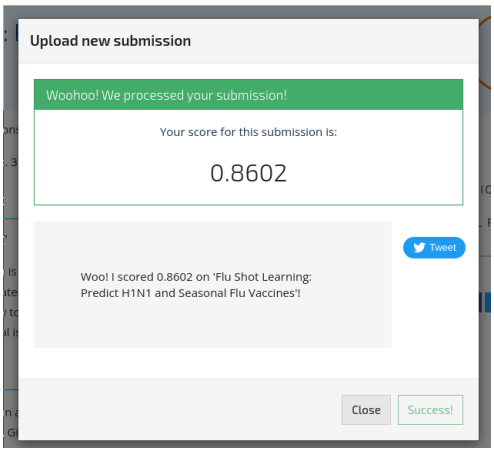
En este caso, empeoramos ligeramente los resultados obtenidos.

3.15. Experimento 14

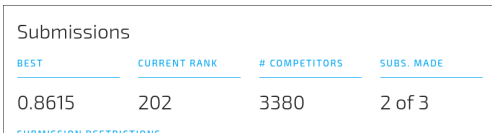
Respecto al procesado o manejo de los datos, no realizamos modificaciones. Respecto al uso de modelos:

1. En vez de usar todos los modelos en el *ensemble*, usamos solo los mejores obtenidos
2. Estos modelos son: *Catboost*, *MLP* y *Xgboost*
3. Por haber dado peores resultados en el anterior experimento, volvemos a usar *softmax* para normalizar los pesos a partir de los resultados en validación

Los resultados se muestran en la siguiente figura:



(a) Score obtenida en la *submission*



(b) Posición de ese momento en la competición

Figura 15: Resultados obtenidos tras hacer la *submission* en la plataforma *online*

Al usar menos modelos en el *ensemble*, a pesar de ser los mejores modelos, vemos que los resultados empeoran significativamente.

4. Conclusiones

En primer lugar, a lo largo de los experimentos ha quedado clara la importancia de realizar un buen procesado de los datos. La normalización de datos, el tipo de imputación de *missing values*, el tipo de borrado de *outliers* o la *numerización* de variables categóricas han sido partes clave en la mejora de los resultados en la competición.

En segundo lugar, hemos usado modelos considerados como *"estado del arte"* a la hora de realizar una tarea de clasificación, como pueden ser *Catboost* o *Xgboost*. Sin embargo, y lo que nos ha proporcionado una mejora constante (aunque no disruptiva) ha sido la construcción apropiada de un *ensemble* con voto suave con todos los modelos explorados hasta el momento.

De hecho, en los últimos experimentos, hemos visto cómo *"adelgazar"* el *ensemble* ha resultado en peores resultados. Incluso cuando hemos quitado del *ensemble* los peores modelos.

A pesar de esto, tenemos que tener en consideración que este proceder para una competición es una buena idea. Sin embargo, para otros escenarios puede no ser ideal. Por ejemplo, podemos considerar los siguientes problemas:

1. Al considerar tantos modelos perdemos mucha capacidad para extraer conocimiento directamente de los modelos. En nuestro caso concreto, predecimos muy bien si una persona se vacuna o no, pero no extraemos información sobre el por qué
2. Los tiempos de entrenamiento, pero sobre todo de inferencia, aumentan considerablemente. Si queremos usar el modelo en entornos de respuesta relativamente rápida, no podremos usar *ensembles* tan grandes

Pasamos ahora a evaluar los resultados. Como ya hemos dicho, por problemas de tiempo, solo realizamos propuestas en la competición hasta el 31 de Diciembre de 2021. Mostramos la posición en la que quedamos en el momento de escribir estas conclusiones:

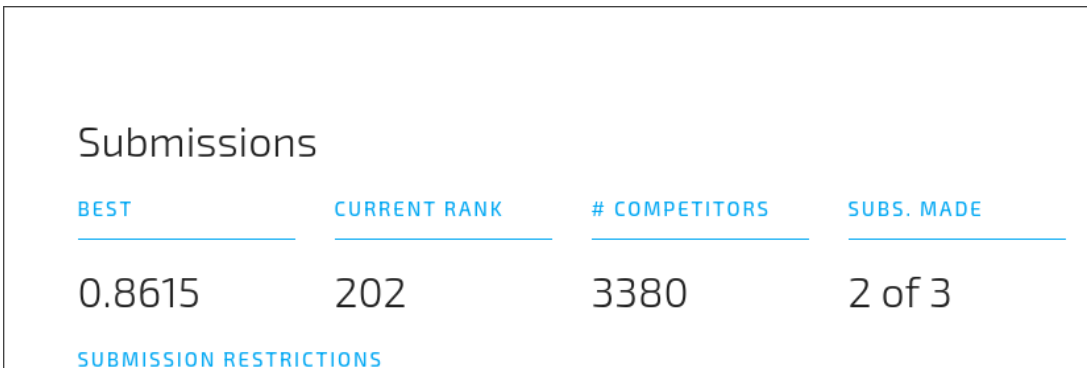


Figura 16: Posición final a la hora de escribir estas conclusiones, el 31 de Diciembre de 2021

Teniendo en cuenta que quedamos en la posición 202 de 3380 competidores, esto supone que quedamos en el mejor 6 % de la competición, en esta fecha. Por inactividad seguramente bajemos muchas posiciones en la competición. Aún así, estamos contentos con los resultados obtenidos. Quedamos en una posición satisfactoria para nosotros. Y además, como demuestra esta memoria, hemos mejorado constantemente los resultados con los experimentos llevados a cabo. Además ha sido una buena oportunidad para trabajar con un problema real, con modelos y técnicas consideradas a día de hoy como *"estado del arte"*.

Y para finalizar, de haber tenido más tiempo para seguir realizando propuestas, seguiríamos probando modelos *"estado del arte"*. Además, seguiríamos trabajando en mejorar el *ensemble*, que nos ha dado una mejora incremental con el aumento de su tamaño. Por ejemplo, probaríamos con modelos como *lightgbm*.

También, y para buscar mejoras de gran calado, buscaríamos mejorar el procesado de datos. Esto ha sido en la práctica más costoso, porque suponía repetir la exploración de parámetros con *Cross Validation* para todos los modelos, lo que nos ha llegado a suponer 14h de procesamiento en *Google Colab*.

5. Referencias

[1] "Competition: Flu shot learning: Predict h1n1 and seasonal flu vaccines." https://www.drivendata.org/competitions/66/flu-shot-learning/page/211/. (Accessed on 12/30/2021).

[2] "Flu shot learning: Predict h1n1 and seasonal flu vaccines - benchmark - drivendata labs." https://www.drivendata.co/blog/predict-flu-vaccine-data-benchmark/. (Accessed on 12/30/2021).