SPRINT 13. ÁRBOLES DE DECISIÓN Y ENSAMBLADOS

PERO ANTES:

ANTES DE NADA: CUADERNO: PAPEL Y BOLI. GRÁFICAS IMPRESAS Y PEGADAS, A VECES, O TABLAS.

- **Conviene tener un cuaderno para cuando se hace un proyecto de ML, se van apuntando notas y conclusiones, hipótesis, números a recordar, etc.**
- **Sirven para volver en el tiempo y comprobar si habías hecho algo de alguna forma o habías hecho suposiciones, además de una gran forma de documentación.**
- **Se apunta desde que significan los acrónimos de las variables.**
- **Qué features son colineales, y por qué me quedo con una en vez de con otra.**
- **Qué features son binarias, a cuales onehot y a cuales ordinal.**
- **Lo equivalente para las numéricas.**
- **Decisiones como elegir un modelo en vez de otro.**
- **Etc, etc.**
- **Por supuesto hipótesis y conclusiones "en rojo".**
- **Bifurcaciones en elecciones.**

Recomendación proyecto de ML:

- **Siempre un miniEDA que seleccione variables con más info aparente (es posible equivocarse y peor performance).**
- **Menos overfitting**
- **Modelo más robusto**
- **Modelo más explicable**

- **Modelo más causal (no correlacional)**
- **Mejor selección de modelo**
- **Hay veces que no es posible.**

Existen aproximaciones distintas:

- **Meter todo y a ver qué sale por fuerza bruta.**
- **Ejemplo ING: variables sin nombre representativo, muy centrado en big-data y nada de data understanding.**
- **Contrapartida modelos causales de procesos industriales, que tenían que ser explicables y tiene que ser validados por ingenieros de procesos de la empresa en la que se está instalando el sistema: tienen que ser efectos conocidos (alguna vez sorpresa contraintuitiva).**

Causalidad (explicabilidad):

- **Que dos cosas correlen no quiere decir que una sea causa de otra.**
- **Además, si lo fueran, ¿cual es causa de cual?**
- **En series temporales se pueden ver qué picos están antes en una variable, por ejemplo, pero no es decisivo, es orientativo.**
- **Se pueden desplazar en el eje temporal y aplicarles pearson, por ejemplo, pero tampoco es definitivo.**
- **Ejemplo: el EDA de Fran sobre el cambio en estilo de juego en la NBA causado por un cambio en un equipo: su pico de puntuación y de anillos estaba antes que los demás equipos.**

SUPERFICIE DE DECISIÓN PARA MODELO LOGÍSTICO CON 4 ENTRADAS.

- **Sólo se pueden ver de 2 en 2 junto con la probabilidad de pertenecer a una clase o a otra.**
- **Los modelos logísticos son una función matemática con curvas suaves y sin cortes.**

- **Son contínuos.**
- **A partir de unos puntos, instancias, se genera la función logística con los parámetros entrenados.**
- **No así los árboles de decisión. Constan de muchas superficies planas puestas unas junto a otras.**
- **Las redes neuronales tienen superficies parecidas a las de la logística, pero más complejas.**

U1. ÁRBOLES DE DECISIÓN E HIPERPARÁMETROS

PREGUNTAS:
Un árbol de decisión sirve para: clasificación y regresión.
Los árboles de decisión son: sencillos de explicar, pero pueden sobreajustar con facilidad.
El conjunto de hiperparámetros y rangos que probamos a la hora de optimizar se llama: Grid.
La medida de impureza de un nodo en un árbol se denomina: índice Gini.
max_depth es un hiperparámetro que sirve para la máxima profundidad en niveles del árbol, puede controlar el overfitting con valores pequeños.

PROBLEMA DE NEGOCIO: MARKETING.

Creamos una caja negra que nos dé los clientes para los que haya más probabilidad de comprar. Esa caja negra tiene una métricas que nos da la confianza.
Ahora creamos la caja negra. ¿Cúal es el target?
Comprar en la campaña de marketing. Una clasificación: si va a comprar o no.
MODELO.predict_proba()
Finalizada la campaña, se mira quienes han comprado y se evalua cómo de bien se ha comportado mi modelo.
Las cajas negras a veces son más complejas y otras más fáciles de explicar.

El objetivo de negocio:

- **¿seleccionar los clientes para la campaña de marketing?.**
- **Queremos hacer la campaña a todos los que digan que sí.**
- **Eso es el recall de la clase que dice sí.**
- **Llamamos sí al 1, no al 0.**

Nos van a dar históricos de campañas de targeting y vamos a presuponer que las campañas del futuro se comportan de forma similar.

En el gridsearch elegiremos la métrica que mejor capture el caso de negocio.

Al contrario que en este caso de negocio, en un posible test de COVID:

- **El objetivo sería la precisión muy alta de la clase negativa**
- **Porque no queremos gente que piense que está sin el virus haciendo vida normal**
- **Los queremos en cuarentena.**

- **De ahí muy importante que no falle en los negativos.**

****

Features elegidas:

- **Categóricas: job (onehot), housing (bin), contact (onehout), outcome (onehot)**
- **Numéricas: edad, duration (log n +1), pdays (log n +2) -> binaria + StandardScaler**

U2 Ensamblado de Modelos: bagging and boosting: random forests y gradient boosting models

CONCEPTOS A REPASAR:

Boosting y Bagging son dos técnicas fundamentales de ensamblaje de modelos utilizadas para obtener mejores rendimientos.

Bagging (Bootstrap Aggregating):

- **En Bagging, múltiples modelos (generalmente del mismo tipo) se entrenan de manera independiente en diferentes subconjuntos de datos creados mediante muestreo aleatorio con reemplazo (bootstrap).**
- Los modelos individuales se entrenan de forma paralela, lo que permite una ejecución más rápida en comparación con métodos secuenciales como Boosting.
- El resultado final se obtiene mediante promedio (en regresión) o votación (en clasificación) de las predicciones de los modelos individuales.
- Ejemplos notables de algoritmos basados en Bagging incluyen Random Forest, que es una colección de árboles de decisión entrenados en diferentes subconjuntos de datos y características.

Boosting:

- **En Boosting, los modelos se entrenan de forma secuencial, y cada modelo subsiguiente intenta corregir los errores de los modelos anteriores.**
- **Durante el entrenamiento, se da más peso a las instancias clasificadas incorrectamente por los modelos anteriores, lo que enfoca el aprendizaje en los ejemplos difíciles.**
- Los modelos débiles se combinan para formar un modelo fuerte.
- Ejemplos de algoritmos basados en Boosting incluyen Gradient Boosting, AdaBoost y XGBoost.

Diferencias:

- **Bagging entrena modelos independientes en paralelo, mientras que Boosting entrena modelos secuencialmente, corrigiendo los errores de los modelos anteriores.**

- En Bagging, cada modelo tiene igual peso en la predicción final, mientras que en Boosting, los modelos se ponderan según su desempeño.
- Boosting tiende a ser más propenso al sobreajuste en comparación con Bagging, ya que los modelos posteriores se ajustan para corregir los errores de los modelos anteriores.
- Boosting a menudo produce modelos más complejos y de alta precisión en comparación con Bagging, pero a costa de un mayor costo computacional y tiempo de entrenamiento.

Ambas técnicas son poderosas y se utilizan ampliamente en la práctica, y la elección entre ellas depende del problema específico y las características del conjunto de datos.

CASO DE NEGOCIO:

- **Desarrollar un predictor de si van a tener diabetes. El mejor predictor posible.**
- **Clasificación.**
- **(Como para el COVID: tiene que ser muy seguro el No por eso de no expandir los contagios. Un falso positivo no es tan problemático)**
- **IMPORTA LA PRECISIÓN DE LA CLASE NEGATIVA, TENER DIABETES.**

PRÁCTICA:

- **Vamos a comparar modelos. Vamos a hacer los pasos habituales de un proyecto de ML: limpiar, train-test, mini-EDA, transformaciones: no porque estamos usando boosting y bagging de DecisionTrees, que no son sensibles a la escala.**
- **Para comparar los modelos con el train set, vamos a hacer un cross_val con la métrica adecuada.**
- **Lo ideal sería hacer un finetunning con GridSearch y después decidir, pero por recursos computacionales no lo hacemos así.**
- **Escogemos un modelo y buscamos los mejores hiperparámetros.**
- **El GridSearch hace un último reentrenamiento en todo el train set una vez decididos los hiperparámetros de la mejor combinación de todo el grid.**
- **El dataset de test sólo se utiliza una vez elegido el modelo y tuneado, si no estaríamos entrenando la selección para el test.**

- **Por tenerlo en código hacemos el ajuste de hiperparámetros de otro modelo.**

MODELOS A PROBAR:

Random Forest es un algoritmo de aprendizaje supervisado utilizado tanto para tareas de clasificación como de regresión. Es una técnica de conjunto que combina múltiples árboles de decisión durante el entrenamiento. Cada árbol individual en el bosque se entrena con una porción aleatoria del conjunto de datos original y realiza predicciones independientes. Luego, las predicciones de cada árbol se combinan mediante votación (en clasificación) o promediado (en regresión) para obtener la predicción final del bosque.

Hiperparámetros tipicos y valores razonables:

1. n_estimators: [50, 100, 200, 300, 400, 500]
2. max_depth: [None, 5, 10, 15, 20]
3. min_samples_split: [2, 5, 10]
4. min_samples_leaf: [1, 2, 4, 8, 16]
5. max_features: ['auto', 'sqrt', 'log2', None]
6. bootstrap: [True, False]

XGBoost (Extreme Gradient Boosting) es una implementación optimizada de Gradient Boosting, que es un algoritmo de aprendizaje supervisado utilizado tanto para tareas de regresión como de clasificación.

Optimización de Gradient Boosting: XGBoost implementa un algoritmo de boosting extremadamente eficiente y escalable, que es una técnica de aprendizaje automático que combina múltiples modelos débiles (generalmente árboles de decisión) para crear un modelo más fuerte

Hiperparámetros tipicos y valores razonables:

1. n_estimators: [50, 100, 200, 300, 400, 500]
2. max_depth: [3, 5, 7, 9, 11]
3. learning_rate: [0.01, 0.05, 0.1, 0.3]
4. subsample: [0.6, 0.7, 0.8, 0.9, 1.0]
5. colsample_bytree: [0.6, 0.7, 0.8, 0.9, 1.0]
6. gamma: [0, 0.1, 0.2, 0.3, 0.4]
7. min_child_weight: [1, 3, 5, 7, 9]
8. reg_lambda (lambda) y reg_alpha (alpha): [0, 0.1, 0.5, 1.0]

max_features = sqrt quiere decir que si tienes un total de **n** características en tu conjunto de datos, el algoritmo Random Forest considerará alrededor de **√n** características al buscar la mejor división en cada nodo del árbol. Esta estrategia se

utiliza para controlar la aleatoriedad y la diversidad entre los árboles en el bosque. Limitar el número de características consideradas en cada división puede ayudar a prevenir el sobreajuste y mejorar la generalización del modelo.

LGBMClassifier (Light Gradient Boosting Machine) es un clasificador basado en árboles de decisión que utiliza el algoritmo Gradient Boosting. Es parte de la biblioteca LightGBM, que es una implementación eficiente y escalable de algoritmos de Gradient Boosting. LightGBM está diseñado para ser rápido y eficiente en el uso de recursos, y es especialmente útil en conjuntos de datos grandes y de alta dimensionalidad.
De Microsoft.
El algoritmo LGBM construye múltiples árboles de decisión débiles de manera secuencial. Cada árbol se construye de forma que minimiza la pérdida del modelo en los datos de entrenamiento.
Hiperparámetros tipicos y valores razonables:

1. n_estimators: [50, 100, 200, 300, 400, 500]
2. max_depth: [-1, 3, 5, 7, 9]
3. learning_rate: [0.01, 0.05, 0.1, 0.3]
4. num_leaves: [20, 30, 40, 50, 60]
5. min_child_samples: [20, 30, 40, 50, 60]
6. subsample: [0.6, 0.7, 0.8, 0.9, 1.0]
7. colsample_bytree: [0.6, 0.7, 0.8, 0.9, 1.0]
8. reg_alpha: [0, 0.1, 0.5, 1.0]
9. reg_lambda: [0, 0.1, 0.5, 1.0]
10. min_child_weight: [1, 3, 5, 7, 9]
11. scale_pos_weight: [1, 2, 3, 4, 5]

Yo no me sé todos los hiperparámetros de todos los modelos. Lo habitual es que cuando estás trabajando con uno en concreto te los mires más en profundidad y los vayas probando y ganando experiencia.

SCORING EN FINETUNNING DE HIPERPARÁMETROS:

Problemas de Clasificación:

1. 'accuracy': Precisión (por defecto). Esta métrica calcula la fracción de muestras correctamente clasificadas.

2. 'precision': Precisión. Es la razón de verdaderos positivos sobre la suma de verdaderos positivos y falsos positivos.
3. 'recall': Recall o sensibilidad. Es la razón de verdaderos positivos sobre la suma de verdaderos positivos y falsos negativos.
4. 'f1': Puntuación F1. Es la media armónica de precisión y recall.
5. 'roc_auc': Área bajo la curva ROC (Receiver Operating Characteristic). Es útil para evaluar la capacidad de discriminación del modelo.
6. 'balanced_accuracy': Precisión equilibrada. Es la media de la sensibilidad (recall) de cada clase.

Problemas de Regresión:

1. 'neg_mean_squared_error': Error cuadrático medio negativo (por defecto). Es el promedio de los cuadrados de los errores.
2. 'neg_mean_absolute_error': Error absoluto medio negativo. Es el promedio de las diferencias absolutas entre las predicciones y los valores verdaderos.
3. 'r2': Coeficiente de determinación R^2. Mide la proporción de la varianza en la variable dependiente que es predecible a partir de las variables independientes.

Estas son algunas de las opciones disponibles en scikit-learn. Además, también puedes definir tus propias funciones de puntuación personalizadas utilizando la función **make_scorer** de la biblioteca **sklearn.metrics**.

Feature Selection:

- **Categóricas: ninguna**
- **Numéricas: preg, plas, mass, pedi, age**