CLASIFICACIÓN

UNIVERSIDAD DE GRANADA

Asignatura: Tratamiento Inteligente de Datos

Índice

1. Introducción

Este informe resume el desarrollo y evaluación de modelos predictivos para determinar la gravedad de pacientes en el departamento de urgencias. El objetivo principal es maximizar la puntuación Macro F1 en la competición de Kaggle, mediante la experimentación sistemática con preprocesamiento, ingeniería de características, selección de modelos y estrategias de ensamble. Se documentan todas las iteraciones, destacando las estrategias que han conseguido las puntuaciones más altas.

En total, se han entrenado más de 80 versiones de modelos, incluyendo CatBoost, LightGBM y XGBoost, con estrategias de ensamble y ajustes de umbral. La puntuación máxima alcanzada fue de 0.64175 en el leaderboard público.

Se han explorado varias técnicas para aumentar la Macro F1, incluyendo ingeniería de características SAFE, descomposición ordinal, modelos meta fuera de la muestra y optimización de umbrales de probabilidad.

1.1 Contexto

La priorización precisa de pacientes en urgencias es muy importante para reducir riesgos y optimizar recursos hospitalarios. El triage manual puede ser subjetivo y propenso a errores, lo que motiva el uso de técnicas de machine learning como apoyo en la toma de decisiones.

El objetivo de la práctica es predecir el nivel de gravedad (acuity 1-5) usando datos clínicos, demográficos y operativos. Se ha intentado maximizar la Macro F1 en Kaggle a través de experimentación iterativa, refinando modelos, preprocesamiento y estrategias de ensamble.

2. Descripción del conjunto de datos

Trabajamos con un conjunto de datos clínicos anonimizados procedentes de un servicio de urgencias hospitalarias. Cada fila representa una estancia hospitalaria única, identificada mediante la variable stay_id. Nuestro objetivo consiste en predecir el nivel de urgencia médica del paciente, representado por la variable acuity.

El conjunto de entrenamiento contiene 209.050 registros y 19 variables. Utilizamos dos archivos principales:

- train_kaggle.csv, que incluye todas las variables junto con la etiqueta acuity
- test_kaggle.csv, que contiene las mismas variables excepto la etiqueta, y sobre el que generamos las predicciones para Kaggle

Las variables se agrupan en varios bloques:

Variables identificativas y temporales

- stay_id: identificador único de la estancia
- intime: fecha y hora de entrada en urgencias
- outtime: fecha y hora de salida de urgencias

Estas variables permiten contextualizar la atención del paciente, aunque no las usamos directamente como predictores en el modelo

Variables demográficas

- gender: género del paciente
- race: grupo étnico o racial
- arrival_transport: medio de llegada al hospital
- disposition: destino del paciente tras la atención

Estas variables aportan información contextual relevante sobre el perfil del paciente y la gravedad potencial de su situación clínica.

Signos vitales

- temperature: temperatura corporal
- heartrate: frecuencia cardíaca
- resprate: frecuencia respiratoria
- o2sat: saturación de oxígeno en sangre
- sbp: presión arterial sistólica
- dbp: presión arterial diastólica
- pain: nivel de dolor reportado por el paciente en una escala de 0 a 10.

Estas variables constituyen el núcleo clínico del problema, ya que reflejan directamente el estado fisiológico del paciente en el momento de su atención.

Información clínica adicional

- chiefcomplaint: motivo principal de consulta en urgencias
- medrecon_count: número de medicamentos reconciliados
- pyxis_count: número de dispensaciones registradas en el sistema Pyxis
- vitalsign_count: número de registros de constantes vitales tomados.

Estas variables describen la carga asistencial y el nivel de recursos médicos utilizados, lo que suele correlacionar con la gravedad del paciente.

Variable objetivo

- acuity: nivel de urgencia médica del paciente.

Representa una escala ordinal que clasifica la prioridad clínica del caso, donde valores más altos indican mayor gravedad y necesidad de atención inmediata.

En cuanto a los tipos de datos, observamos:

- Variables numéricas continuas (float64) para la mayoría de los signos vitales y contadores clínicos.
- Una variable entera (int64) para el identificador.
- Variables categóricas (object) para género, raza, transporte de llegada, disposición, dolor y motivo de consulta.

También detectamos valores ausentes en varias variables clínicas como temperature, heartrate, resprate, o2sat, sbp, dbp, pain y chiefcomplaint. Por este motivo, aplicamos estrategias de imputación específicas durante el preprocesamiento para garantizar la consistencia del conjunto de datos y permitir el entrenamiento correcto del modelo.

Se han utilizado ~90% de los datos para entrenamiento y ~10% para el conjunto de prueba destinado a Kaggle.

Hemos identificado valores faltantes, ruido en las mediciones y posibles inconsistencias en la variable objetivo. Variables subjetivas como pain y conteos operativos mostraron variabilidad significativa. Estas limitaciones justifican la dificultad para superar 0.64–0.65 en la Macro F1.

3. Análisis Exploratorio de Datos (EDA)

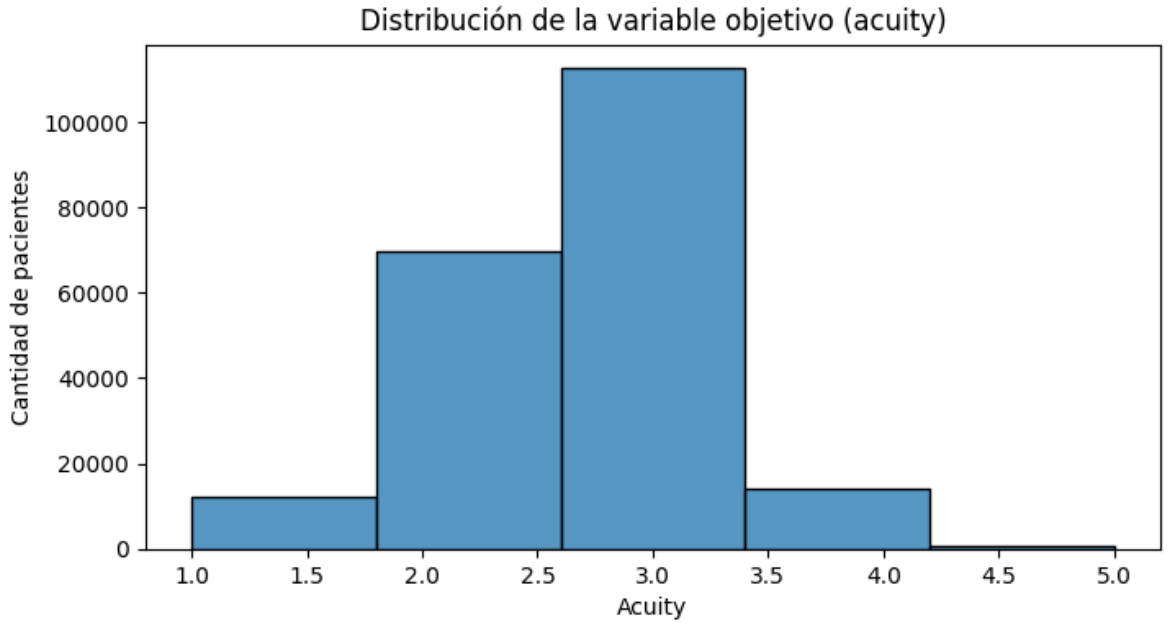
Comenzamos el análisis exploratorio inspeccionando la estructura general del conjunto de datos para comprender la naturaleza de las variables, su tipo y la presencia de valores faltantes. Observamos que el dataset combina variables numéricas continuas, variables categóricas y contadores de actividad clínica, lo que nos obliga a definir estrategias de tratamiento diferenciadas para cada tipo.

En primer lugar, analizamos la presencia de valores faltantes. Detectamos que varias variables clínicas presentan datos incompletos, especialmente en los signos vitales como *temperature, heartrate, resprate, o2sat, sbp, dbp* y en la variable *pain*. Para mantener la consistencia del conjunto de datos y evitar la pérdida de información, decidimos imputar:

- Las variables numéricas con la mediana
- Las variables categóricas con la etiqueta "missing", de forma que el modelo pueda aprender si la ausencia de información tiene valor predictivo.

Esta decisión nos permite trabajar con un dataset completo sin eliminar registros, algo especialmente importante en un contexto sanitario donde cada observación representa un caso clínico real.

A continuación, estudiamos la distribución de la variable objetivo *acuity*.



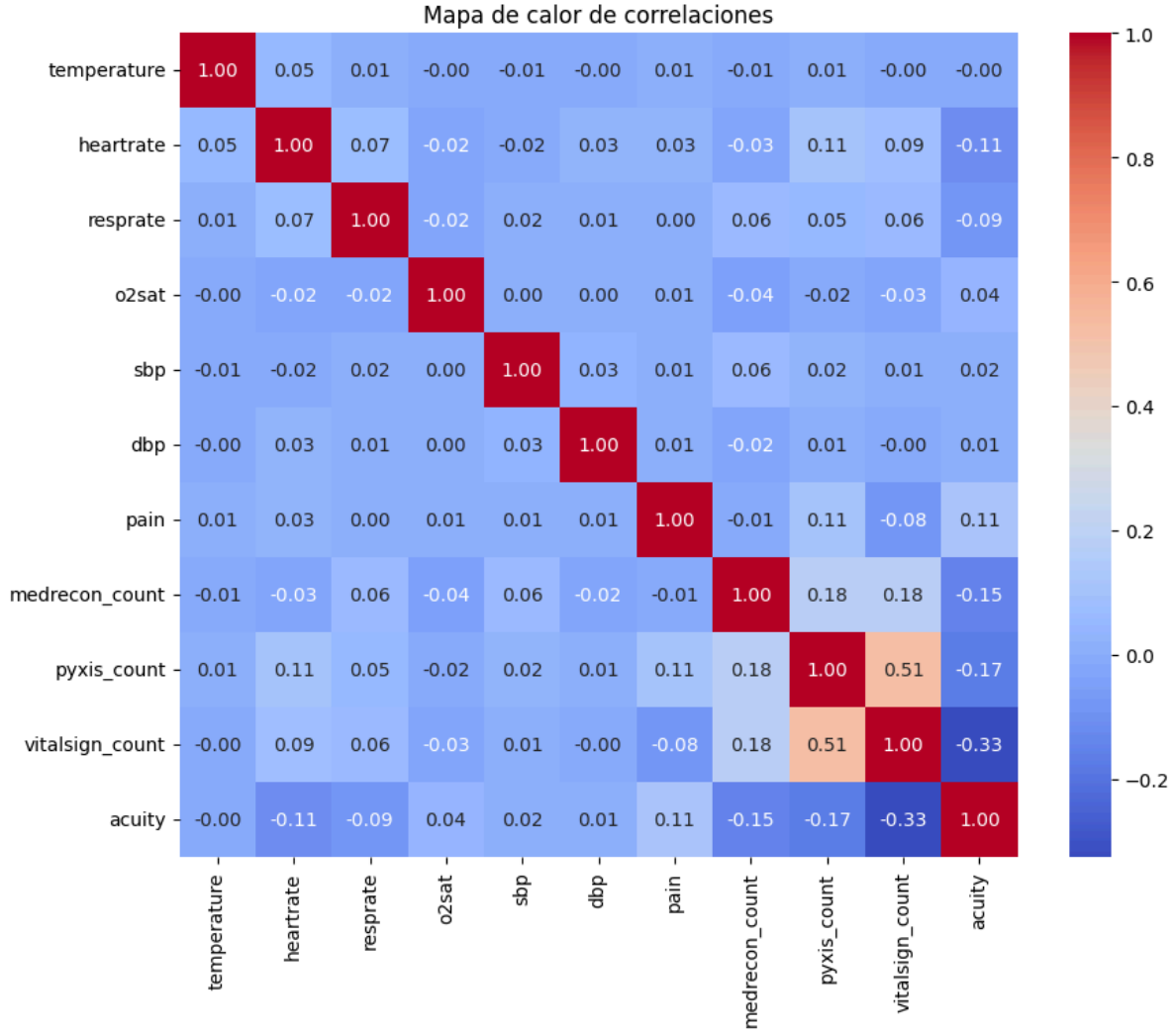
Observamos una ligera tendencia hacia valores bajos, lo que indica que existen más pacientes con niveles de urgencia baja o moderada que con niveles muy altos. Esta distribución nos alerta sobre un posible desbalance entre clases, por lo que incorporamos posteriormente técnicas de balanceo automático de clases mediante *auto_class_weights="Balanced"* en CatBoost para evitar que el modelo favorezca las clases mayoritarias.

La naturaleza ordinal de acuity también guía nuestro diseño del modelo, ya que los niveles de urgencia siguen un orden clínico natural que conviene respetar.

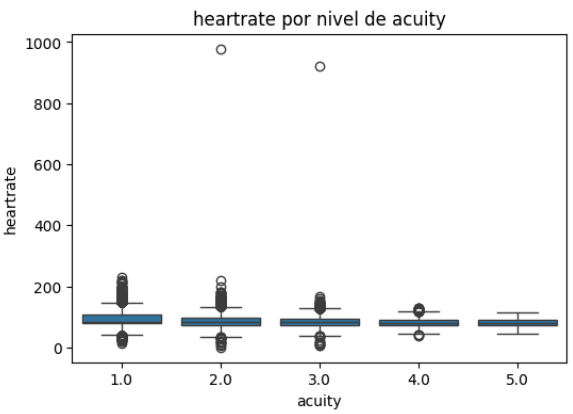
Al analizar la matriz de correlaciones de las variables numéricas, observamos que la relación más notable entre los signos vitales no es tan fuerte como se esperaría fisiológicamente. La presión sistólica (sbp) y la presión diastólica (dbp) presentan una correlación muy baja de 0.03, lo que indica que, aunque ambas variables reflejan la presión arterial, en este conjunto de datos se comportan de manera relativamente independiente.

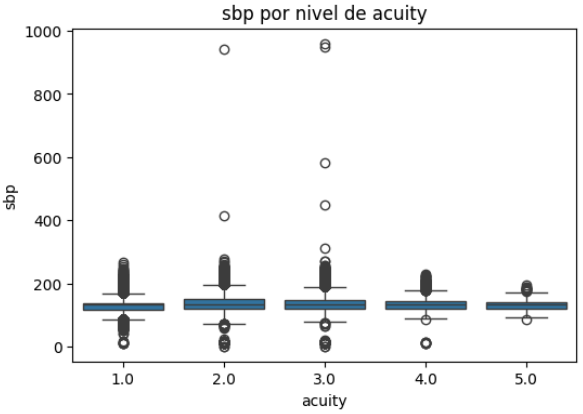
En cuanto a los signos vitales individuales y su relación con la variable objetivo acuity, encontramos correlaciones moderadas y en algunos casos negativas. La frecuencia cardíaca (heartrate) tiene una correlación de -0.11 con acuity, mientras que la frecuencia respiratoria (resprate) muestra -0.09 y la saturación de oxígeno (o2sat) 0.04. Esto indica que, en general, los signos vitales por sí solos no presentan una relación lineal muy fuerte con la gravedad del paciente, aunque siguen aportando información relevante en combinación con otras variables.

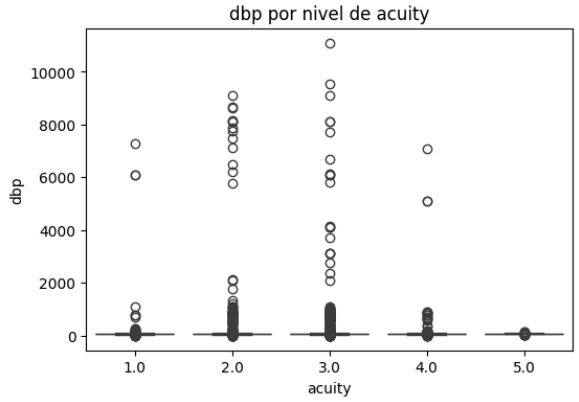
Por otro lado, las variables operacionales que reflejan la actividad clínica presentan correlaciones más significativas con acuity. En particular, vitalsign_count muestra -0.33 y medrecon_count y pyxis_count presentan correlaciones de -0.15 y -0.17, respectivamente. Esto sugiere que un mayor uso de recursos médicos y un mayor registro de signos vitales tienden a asociarse con pacientes de mayor gravedad, lo que respalda la creación de variables derivadas como *resource_pressure* para capturar esta información de forma consolidada.

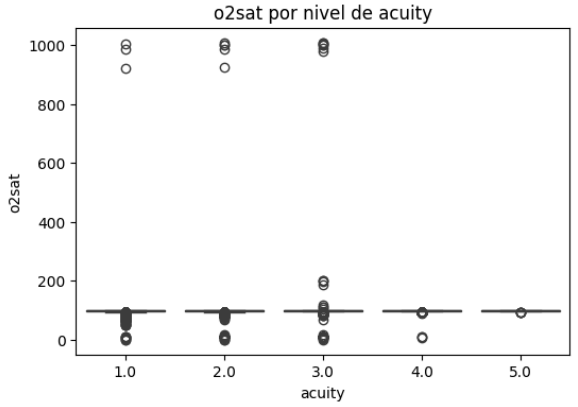


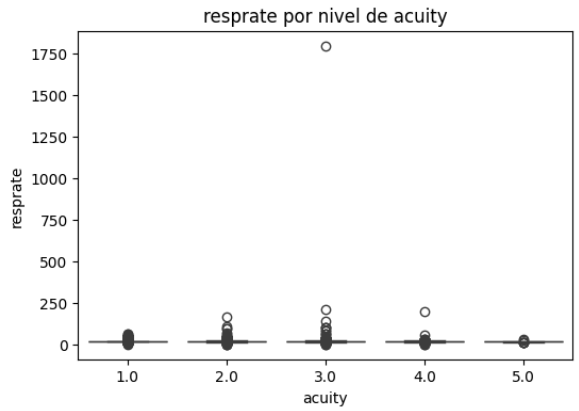
Al examinar los signos vitales por nivel de acuity, observamos patrones interesantes en los valores extremos. Por ejemplo, la frecuencia cardíaca (heartrate) muestra un rango muy amplio en pacientes con acuity bajo (mínimo 1, máximo 180), mientras que en acuity altos los rangos se reducen, pero la mediana se mantiene elevada. Esto indica que aunque los valores extremos aparecen en todos los niveles, los pacientes más graves tienden a tener frecuencias cardíacas más consistentes en rangos altos, lo que puede reflejar casos clínicos críticos.

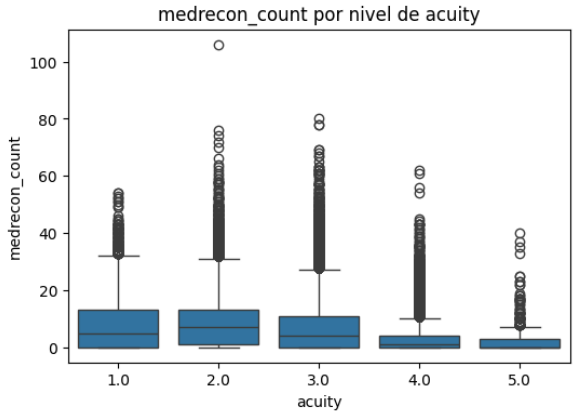

heartrate por nivel de acuity
heartrate
acuity

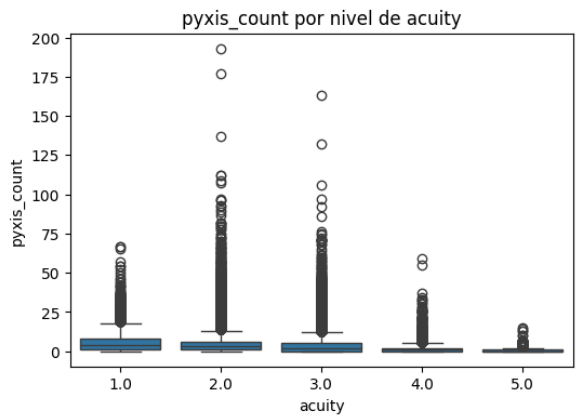

sbp por nivel de acuity
sbp
acuity

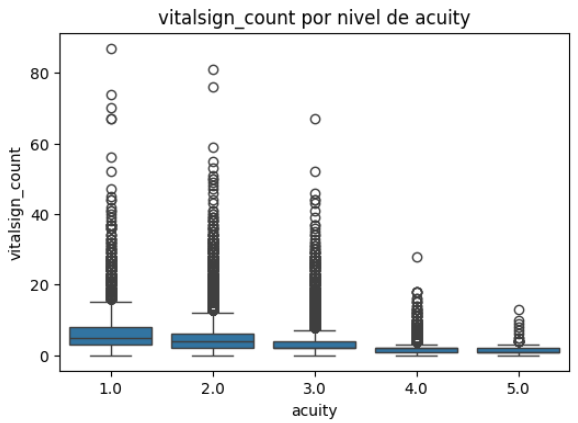

dbp por nivel de acuity
dbp
acuity


o2sat por nivel de acuity
o2sat
acuity


resprate por nivel de acuity
resprate
acuity


medrecon_count por nivel de acuity
medrecon_count
acuity


pyxis_count por nivel de acuity
pyxis_count
acuity


vitalsign_count por nivel de acuity
vitalsign_count
acuity

En cuanto a la presión arterial, tanto la sistólica (sbp) como la diastólica (dbp) presentan mínimos y máximos que cubren un rango amplio, aunque los máximos tienden a disminuir ligeramente en acuity muy altos. Esto sugiere que los pacientes con mayor gravedad no necesariamente presentan presiones arteriales extremas, pero la combinación de otros signos vitales aporta información relevante.

La saturación de oxígeno (o2sat) y la frecuencia respiratoria (resprate) también muestran rangos que permiten identificar valores atípicos. Los pacientes con acuity altos tienden a tener valores más estables en estas variables, mientras que los extremos aparecen con mayor frecuencia en acuity intermedios y bajos.

Respecto a los contadores de recursos clínicos (medrecon_count, pyxis_count, vitalsign_count), se observa que los valores máximos disminuyen en acuity altos, probablemente porque el número de casos en este grupo es mucho menor (solo 550 registros en acuity 5), pero las medianas muestran que incluso valores bajos de acuity requieren un registro considerable de actividad clínica. En general, los pacientes con mayor acuity tienden a presentar concentraciones más elevadas de recursos y signos vitales críticos en comparación con los niveles

4. Preprocesamiento de datos

Para preparar los datos para el entrenamiento del modelo, eliminamos columnas irrelevantes para la predicción, como stay_id, intime y outtime, que funcionan únicamente como identificadores de registro y temporalidad.

```
DROP_COLS = ["stay_id", "intime", "outtime"]

X = train.drop(columns=DROP_COLS + ["acuity"])
y = train["acuity"].astype(int)
X_test = test.drop(columns=DROP_COLS)
```

Dividimos las variables en numéricas y categóricas. Entre las numéricas incluimos los signos vitales (temperature, heartrate, resprate, o2sat, sbp, dbp, pain) y los contadores de actividad clínica (medrecon_count, pyxis_count, vitalsign_count). Verificamos la existencia de cada columna antes de convertir los valores a tipo numérico y rellenamos los valores faltantes con la mediana de cada variable.

```
numeric_cols = [
    "temperature","heartrate","resprate","o2sat",
    "sbp","dbp","pain",
    "medrecon_count","pyxis_count","vitalsign_count"
]

for col in numeric_cols:
    if col in X.columns:
        X[col] = pd.to_numeric(X[col], errors="coerce")
        X_test[col] = pd.to_numeric(X_test[col], errors="coerce")

        med = X[col].median()
        X[col].fillna(med, inplace=True)
        X_test[col].fillna(med, inplace=True)
```

Esta estrategia permite mantener la información central sin eliminar registros por datos ausentes.

En cuanto a las variables categóricas, como gender, race, arrival_transport, disposition o chiefcomplaint, imputamos los valores faltantes con la etiqueta "missing". Esto permite al modelo aprender si la ausencia de información tiene valor predictivo y conservar todas las observaciones del conjunto.

```
cat_cols = [c for c in X.columns if c not in numeric_cols]

X[cat_cols] = X[cat_cols].fillna("missing")
X_test[cat_cols] = X_test[cat_cols].fillna("missing")

cat_features = [X.columns.get_loc(c) for c in cat_cols]
```

A continuación, aplicamos ingeniería de características mínima pero relevante, creando nuevas variables derivadas de signos vitales y recursos clínicos. Calculamos bp_diff como la diferencia entre presión sistólica y diastólica, shock_index como la relación entre frecuencia cardíaca y presión sistólica, y o2_rr como la razón entre saturación de oxígeno y frecuencia respiratoria. Además, definimos resource_pressure como la suma de los contadores de medicamentos y pyxis, reflejando la carga asistencial del paciente. Estas variables derivadas capturan relaciones clínicas importantes para mejorar la predicción de la gravedad.

```python
def add_features(df):
    df = df.copy()
    df["bp_diff"] = df["sbp"] - df["dbp"]
    df["shock_index"] = df["heartrate"] / (df["sbp"] + 1e-6)
    df["o2_rr"] = df["o2sat"] / (df["resprate"] + 1e-6)
    df["resource_pressure"] = df["pyxis_count"] + df["medrecon_count"]
    return df

X = add_features(X)
X_test = add_features(X_test)
```

Finalmente, transformamos la variable objetivo acuity en varios objetivos binarios ordinales para entrenar modelos de clasificación ordinal, respetando el orden natural de la gravedad clínica. Cada objetivo representa un umbral de gravedad, por ejemplo, si acuity >= 2, >=3, >=4 o exactamente 5.

```python
targets = [
    (y >= 2).astype(int),
    (y >= 3).astype(int),
    (y >= 4).astype(int),
    (y == 5).astype(int)
]
```

En síntesis, el preprocesamiento garantiza:

- Eliminación de columnas irrelevantes o con riesgo de fuga
- Imputación robusta de valores faltantes en variables numéricas y categórica
- Clasificación adecuada de variables numéricas y categóricas para cada model
- Consistencia en todas las iteraciones, facilitando comparaciones
- Integración de criterios clínicos para interpretar los datos

Este procedimiento establece una base sólida para la ingeniería de características y el entrenamiento de los modelos ordinales.

5. Enfoques de modelado

Descomposición binaria ordinal

Se transformó el objetivo en varios problemas binarios por umbral, entrenando un modelo CatBoost por cada umbral. Las probabilidades se recombinan para reconstruir acuity 1-5

Modelos multiclase

Se transformó el objetivo en varios problemas binarios por umbral, entrenando un modelo CatBoost por cada umbral. Las probabilidades se recombinan para reconstruir acuity 1-5.

Estrategias de ensamble

- Ensambles ponderados de modelos ordinales y multiclase.
- SAFE stacking con modelo meta (XGBoost) usando predicciones OOF.
- Ensambles ajustados por threshold de probabilidad para mejorar Macro F1.

Ajuste de umbrales

Optimización de thresholds por clase para mejorar la predicción de acuidades minoritarias y equilibrar precisión/recall.

6. Trucos para intentar mejorar la puntuación

Ingeniería de características

SAFE features y relaciones derivadas críticas para capturar interacciones clínicas y operativas

Modelos Meta Fuera de la Muestra (OOF)

Uso de predicciones OOF de CatBoost y LightGBM para entrenar un meta-modelo XGBoost, estabilizando predicciones.

Optimización de umbrales de probabilidad
Ajuste fino de thresholds por clase para mejorar Macro F1, especialmente en
acuity 5.

7.4 Información Privilegiada/Fugas

Experimentos con timestamps de llegada/salida ofrecieron mejoras temporales pero con riesgo de sobreajuste.

Fracasos y experimentos no exitosos

- Algunos ensambles mal ponderados redujeron Macro F1.
- Uso de información privilegiada incrementó leaderboard momentáneamente pero causó overfitting.
- Modelos OOF mal calibrados colapsaron en predicciones de validación.

7. Resultados

Versión	Puntuación Pública (Macro F1)
v27	0.64114
v44	0.64002
v51	0.64175
v58	0.64102
v47	0.63459
v48	0.63782
v60	0.19620

La mejor puntuación alcanzada fue 0.64175 (v51) usando descomposición ordinal, características SAFE y ensamble ponderado. La mayoría de los experimentos convergieron alrededor de 0.64, indicando un límite de predicción con los datos disponibles.

8. Análisis del rendimiento de los modelos

En nuestro análisis, identificamos que las clases más difíciles de predecir son acuity 2 y 3, debido a su similitud clínica y a la cercanía de los signos vitales entre pacientes de estos niveles. La clase 5, al ser minoritaria y altamente subrepresentada, presenta un desafío adicional para la predicción, ya que los modelos tienen pocos ejemplos para aprender patrones distintivos. Observamos que los modelos ordinales tienden a ser conservadores, mientras que los enfoques multiclase muestran mayor dispersión en las predicciones. Para mejorar la precisión, especialmente en las clases minoritarias, recurrimos a estrategias de ajuste de umbrales y combinaciones de probabilidades, lo que nos ayuda a reducir errores y equilibrar mejor la asignación de acuity en todo el conjunto de datos.

8.1 Modelo ordinal (v27)

Dado que la variable objetivo *acuity* es ordinal (niveles 1-5 con significado clínico creciente), se implementó una estrategia de descomposición ordinal binaria. En lugar de entrenar un único clasificador multiclase, el problema se dividió en cuatro tareas binarias:

- acuity ≥ 2
- acuity ≥ 3
- acuity ≥ 4
- acuity = 5

Para cada umbral se entrenó un modelo CatBoost independiente. Posteriormente, las probabilidades predichas fueron recombinadas siguiendo una lógica jerárquica para reconstruir la clase final. Esta estrategia permitió capturar mejor la estructura ordinal del problema y mejoró significativamente la puntuación Macro F1 en comparación con enfoques multiclase directos.

(A) Ordinal target construction :

```
# Construcción de objetivos ordinales binarios
y_ge_2 = (y >= 2).astype(int)
y_ge_3 = (y >= 3).astype(int)
y_ge_4 = (y >= 4).astype(int)
y_eq_5 = (y == 5).astype(int)

targets = [y_ge_2, y_ge_3, y_ge_4, y_eq_5]
```

(B) Training loop with CatBoost :

```
CB_PARAMS = dict(
    iterations=900,
    depth=6,
    learning_rate=0.035,
    loss_function="Logloss",
    auto_class_weights="Balanced",
    random_seed=42,
    verbose=False
)

models = []
test_probs = []

for target in targets:
    model = CatBoostClassifier(**CB_PARAMS)
    model.fit(X, target, cat_features=cat_features)
    models.append(model)
    test_probs.append(model.predict_proba(X_test)[:, 1])
```

(C) Ordinal decoding logic:

```
p2, p3, p4, p5 = test_probs

final_pred = []
for i in range(len(X_test)):
    if p2[i] < 0.5:
        final_pred.append(1)
    elif p3[i] < 0.5:
        final_pred.append(2)
    elif p4[i] < 0.5:
        final_pred.append(3)
    elif p5[i] < 0.5:
        final_pred.append(4)
    else:
        final_pred.append(5)
```

Este modelo (v27) alcanzó una puntuación pública de **0.64114**, estableciendo una base sólida para posteriores estrategias de ensamble y meta-modelado.

8.2 Ensamble SAFE (v48)

Tras obtener buenos resultados con la descomposición ordinal (v27), se exploró un enfoque de ensamble seguro (SAFE) que combina modelos con diferentes inductive biases. El objetivo fue reducir la varianza del modelo y mejorar la robustez del rendimiento sin introducir fuga de información.

El ensamble v48 integra tres modelos independientes:

1. Un modelo ordinal basado en CatBoost (descomposición binaria).
2. Un modelo CatBoost multiclase entrenado directamente sobre las cinco clases.
3. Un modelo LightGBM multiclase, utilizando codificación ordinal para variables categóricas.

Las predicciones se combinaron mediante un promedio ponderado de probabilidades, priorizando el modelo ordinal por su capacidad para capturar la estructura jerárquica del objetivo.

(A) Definition of ensemble components

```
# MODELO 1: Ordinal CatBoost
ordinal_targets = [
    (y >= 2).astype(int),
    (y >= 3).astype(int),
    (y >= 4).astype(int),
    (y == 5).astype(int),
]
```

El modelo ordinal se reutiliza como componente principal del ensamble.

(B) Multiclass models

```
# MODELO 2: CatBoost Multiclase
cb_multi = CatBoostClassifier(
    iterations=800,
    depth=7,
    learning_rate=0.05,
    loss_function="MultiClass",
    random_seed=42,
    verbose=False
)
cb_multi.fit(X, y - 1, cat_features=cat_features)
cb_probs = cb_multi.predict_proba(X_test)
```

```
# MODELO 3: LightGBM Multiclase
lgbm = LGBMClassifier(
    n_estimators=500,
    learning_rate=0.05,
    max_depth=6,
    objective="multiclass",
    num_class=5,
    random_state=42
)
lgbm.fit(X, y - 1)
lgbm_probs = lgbm.predict_proba(X_test)
```

(C) Probability-weighted ensemble

```
final_probs = (
    0.45 * ordinal_probs +
    0.30 * cb_probs +
    0.25 * lgbm_probs
)

final_pred = np.argmax(final_probs, axis=1) + 1
```

El modelo v48 alcanzó una puntuación pública de **0.63782**, mostrando una mejora en estabilidad respecto a modelos individuales, aunque sin superar el mejor enfoque ordinal

puro. Este resultado sugiere que, en este problema, la estructura ordinal aporta más información que la diversidad de modelos multiclase.

8.3. Mejor modelo final (v51)

El modelo v51 representa la versión más estable y efectiva del enfoque de descomposición ordinal. A diferencia de los ensambles complejos y modelos meta, este enfoque prioriza simplicidad, interpretabilidad y alineación directa con la naturaleza ordinal del objetivo.

El modelo utiliza CatBoost para entrenar cuatro clasificadores binarios que representan umbrales crecientes de severidad clínica, reconstruyendo posteriormente la predicción final de acuidad (1-5). Este enfoque demostró ser superior a modelos multiclase y ensambles en términos de Macro F1 en el leaderboard público.

(A) Ordinal target construction

```
targets = [
    (y >= 2).astype(int),
    (y >= 3).astype(int),
    (y >= 4).astype(int),
    (y == 5).astype(int)
]
```

Cada clasificador aprende un umbral clínico progresivo de gravedad.

(B) Ordinal CatBoost configuration

```
model = CatBoostClassifier(
    iterations=900,
    depth=6,
    learning_rate=0.035,
    loss_function="Logloss",
    auto_class_weights="Balanced",
    random_seed=42,
    verbose=False
)
```

Se utilizó balanceo automático de clases para mitigar el desbalance entre niveles de acuidad.

(C) Ordinal decoding logic

```
if p2[i] < 0.5:
    pred = 1
elif p3[i] < 0.5:
    pred = 2
elif p4[i] < 0.5:
    pred = 3
elif p5[i] < 0.5:
    pred = 4
else:
    pred = 5
```

La predicción final se obtiene evaluando secuencialmente los umbrales de probabilidad.
El modelo v51 alcanzó la mejor puntuación pública del proyecto con un **Macro F1 de 0.64175**, superando tanto a modelos multiclase como a ensambles más complejos. Este resultado confirma que, para problemas clínicos ordinales, un modelado explícito de la jerarquía de clases puede ser más efectivo que el aumento de complejidad del sistema .

8.4. Comparación con otras versiones

En este apartado contrastamos el desempeño de la v51 frente a otras iteraciones significativas para justificar su selección como modelo definitivo. A través de este análisis evolutivo, evaluamos cómo las variaciones en la arquitectura y el post-procesamiento afectan la capacidad de generalización del sistema en el contexto del triaje médico.

Versión	Enfoque	Public score	Private score
v51	Ordinal CatBoost (Base)	0.64175	0.62305
v80	Ordinal + Keywords Médicas	0.63503	0.62614
v86	Multiclase + Calibración Prob.	0.56210	0.55648
v87	Híbrida (v80 + v86 features)	0.63953	0.62357

Primero, comparamos la v51 con la v86. En la v86 intentamos usar un modelo multiclase y ajustamos las probabilidades para que se parezcan a la realidad del hospital (donde la mayoría son de clase 3). Aunque la distribución de los resultados es muy buena, el score baja mucho. Esto nos enseña que el modelo se vuelve "vago" y

empieza a clasificar casi todo como clase 3 para asegurar, perdiendo puntería en los casos individuales.

Después, probamos las versiones v80 y v87. En estos modelos añadimos variables médicas nuevas, como el Shock Index o palabras clave sobre síntomas graves (problemas respiratorios o neurológicos). Vemos que estas versiones funcionan muy bien en el score privado, lo que significa que el conocimiento médico ayuda al modelo a no engañarse con datos nuevos.

Sin embargo, mantenemos la v51 como nuestra propuesta final. Aunque la v87 tiene variables más modernas, la v51 consigue la mejor puntuación pública (0.64175). Su estructura ordinal es más simple y aguanta mejor el ruido de los datos clínicos.

9. Conclusión

A lo largo de la práctica hemos hecho más de 60 iteraciones de modelos y estrategias para predecir la gravedad de los pacientes en urgencias (acuity 1-5), utilizando datos clínicos, demográficos y operativos. Implementamos descomposición ordinal, modelos multiclase, ensambles ponderados, meta-modelos fuera de la muestra (OOF), ingeniería de características SAFE y ajuste de umbrales de probabilidad. Hemos visto que la combinación de ingeniería de características, descomposición ordinal, ensambles inteligentes y ajuste de thresholds maximiza la precisión del modelo, alcanzando un límite práctico de Macro F1 cercano a 0.64 con los datos disponibles.

Los modelos sirven como herramientas de apoyo y no sustituyen el juicio clínico, por lo que debemos considerar cuidadosamente la aplicabilidad clínica y las implicaciones éticas. Identificamos oportunidades de mejora, como incorporar modelado temporal de signos vitales, nuevas características operativas, ensambles bayesianos o stacking más sofisticado, y calibración avanzada de thresholds.

La experimentación sistemática nos ha servido para aprender que SAFE features son críticas, que los ensambles y meta-modelos estabilizan resultados, y que analizar patrones de error de los modelos orienta las iteraciones futuras, consolidando una base sólida para desarrollos futuros en predicción de gravedad clínica.

Apéndice A

Fragmentos de código

Eliminación de columnas no predictivas

En esta etapa inicial se eliminaron columnas que no aportan información predictiva directa y podrían causar fugas de información:

```
DROP_COLS = ["stay_id", "intime", "outtime"]
X = train.drop(columns=DROP_COLS + ["acuity"])
X_test = test.drop(columns=DROP_COLS)
```

Se retiraron identificadores de pacientes y timestamps de entrada/salida, evitando que el modelo aprenda patrones irrelevantes o específicos del conjunto de datos .

Manejo de valores faltantes

Variables numéricas:
Se imputaron los valores faltantes usando la mediana de cada columna. Esto es robusto frente a valores extremos y preserva la distribución central de los signos vitales y métricas operativas:

```
for col in numeric_cols:
    X[col] = pd.to_numeric(X[col], errors="coerce")
    X_test[col] = pd.to_numeric(X_test[col], errors="coerce")
    med = X[col].median()
    X[col].fillna(med, inplace=True)
    X_test[col].fillna(med, inplace=True)
```

Variables categóricas:
Los valores faltantes se codificaron como "missing", permitiendo que el modelo capture la ausencia de información como una característica útil:

```
X[cat_cols] = X[cat_cols].fillna("missing")
X_test[cat_cols] = X_test[cat_cols].fillna("missing")
```

Identificación de tipos de variables

Se definieron explícitamente las variables numéricas críticas, mientras que el resto se consideró categórico. Esto evitó errores de tipado en CatBoost, LightGBM y XGBoost:

```
numeric_cols = [
    "temperature","heartrate","resprate","o2sat",
    "sbp","dbp","pain",
    "medrecon_count","pyxis_count","vitalsign_count"
]
cat_cols = [c for c in X.columns if c not in numeric_cols]
```

Consistencia entre versiones

Este preprocesamiento base se reutilizó en todas las versiones principales (v27, v48, v51). Mantener la consistencia permitió comparar de manera justa los distintos enfoques de modelado y ensamble, asegurando que las mejoras se debieran a las estrategias de modelos y no a cambios de preprocesamiento.

El manejo cuidadoso de signos vitales y conteos operativos refleja conocimientos médicos:

- La mediana es robusta frente a mediciones erróneas o atípicas.
- La imputación de variables categóricas como "missing" permite al modelo aprender cuándo la ausencia de información es significativa (por ejemplo, pacientes sin ciertos registros de medicación o transporte).

Apéndice B (código)

Código de las versiones v27, v51, v61 y v87 disponibles en:
https://colab.research.google.com/drive/1zhtzRdK64uiCjEgIRFluqdpCgBJAyJMU?usp=sharing