UPB

Documentación de Sistema de predicción de estrés estudiantil y generación de acompañamiento personalizado

Semillero A+D

I. Introducción general al proyecto

Este proyecto implementa un sistema integral de detección y acompañamiento del estrés en estudiantes. El proceso inicia con un predictor de estrés basado en variables académicas, financieras y personales (`matriz_1.py`), cuyo resultado alimenta un generador de contenido personalizado mediante IA (`API.py`). Finalmente, se habilita un módulo de chat interactivo (`chat.py`) que utiliza el plan sugerido por la IA para guiar al estudiante en tiempo real. El flujo de trabajo es secuencial: los datos del estudiante ingresan al predictor de estrés, el cual produce puntuaciones y causas de estrés en un archivo Excel; este archivo se usa luego para generar planes de apoyo personalizados mediante el modelo generativo Gemini `(gemini-2.0-flash)` .; por último, el estudiante puede entablar un chat usando el plan generado como contexto inicial. Cada módulo se comunica mediante archivos Excel para facilitar la interoperabilidad y la revisión manual de los datos, el cual en un futuro seria reemplazado por una base de datos propia de la universidad.

II. Descripción de los módulos y su relación

- `matriz_1.py` **(Predictor de estrés)**: Lee datos académicos y financieros de cada estudiante (por ejemplo GPA, asistencias, situación de beca, deudas, etc.) desde un archivo CSV o Excel. Calcula variables de riesgo estandarizando cuantitativas y transformando indicadoras. Combina estos factores según pesos heurísticos para obtener una puntuación continua de estrés. Posteriormente asigna niveles discretos (1–5) y etiqueta la causa principal (económica, motivacional o emocional) según sub-puntuaciones normalizadas. Finalmente guarda un nuevo Excel con las columnas originales más `STRESS_SCORE`, `STRESS_LEVEL` y `STRESS_CAUSE`.

- `API.py` **(Generador de acompañamiento personalizado)**: Carga el archivo de estrés generado, toma los primeros estudiantes, (ejemplo: fila superior) y para cada uno construye un *prompt* que incorpora su nombre, nivel de estrés, causas identificadas y descripción de la carrera. Este *prompt* se envía a la API del modelo Gemini para generar texto de orientación personalizado. El texto de respuesta se guarda junto con la información del estudiante en un nuevo archivo Excel de resultados de acompañamiento (`resultados_acompanamiento_final.xlsx`). Periódicamente guarda progresos parciales para evitar pérdidas.
- `chat.py` **(Chat interactivo en tiempo real)**: Carga el Excel de planes de acompañamiento generado por `API.py`. El usuario ingresa el ID de su estudiante y el módulo selecciona el plan correspondiente. Antes de iniciar el chat, se realiza una pequeña encuesta automática con consejos genéricos para sondear el estado del estudiante. Luego inicia una conversación con Gemini: se 'siembra' en el historial del chat el plan personalizado como contexto inicial, luego el estudiante puede escribir preguntas o comentarios, y Gemini responde como un tutor experto siguiendo el plan dado. Así se logra un acompañamiento en tiempo real usando el plan precomputado.

La relación entre módulos es lineal: el predictor de estrés produce datos que alimentan el generador de acompañamiento, y éste a su vez provee el contexto para el módulo de chat. Usar archivos Excel como medio de intercambio garantiza formato tabular estándar y facilita la inspección manual o integración con otras herramientas por el momento.

III. Justificación de decisiones de diseño

A. Selección de variables y ponderaciones heurísticas

Las variables usadas (GPA, asistencias, horas de estudio, situación financiera, etc.) se eligieron por su relevancia en la literatura sobre estrés académico. Por ejemplo, un bajo *GPA* y altas deudas son indicadores clásicos de riesgo académico y financiero. Cada factor de riesgo recibió un peso heurístico (por ejemplo, `GPA_RISK` con peso 0.70, `DEBTS` con 0.80) basado en la experiencia de dominio y se normalizó mediante Z-score cuando es continuo o transformaciones inversas para booleanos (por ejemplo, `CAREER_RISK` = 1 - puntos de carrera). Estos pesos reflejan la importancia relativa: se da más peso a indicadores críticos (GPA, deudas) y menos a factores menores. Esta elección ponderada permite acentuar factores determinantes en la puntuación de estrés y es compatible con técnicas de construcción de

índices compuestos.

B. Normalización y saturación de la puntuación

Las variables continuas (horas estudiadas, tareas por semana, tiempo de trayecto, pensión) se normalizan usando *Z-score* para centrar en cero y escalar según la desviación estándar. Es decir, cada valor X se transforma mediante:

$$Z = \frac{X - \mu}{\sigma} \tag{1}$$

tal como implementa el código `matriz_1.py`. Esto evita que variables con unidades o escalas muy distintas dominen el índice. Además, al combinar los factores de riesgo se suma un *bias* inicial fijo (BIAS=1.5) y luego se aplica saturación: la puntuación cruda se limita a un máximo de 4.0 antes de sumar 1 para llevarla al rango 1–5. Matemáticamente:

$$\text{score\_crudo} = \text{BIAS} + \sum_i w_i R_i, \quad \text{STRESS\_SCORE} = \min(\text{score\_crudo}, 4{,}0) + 1{,}0 \tag{2}$$

La saturación evita que valores extremos sobrepasen el nivel máximo de estrés (5) y mejora la robustez ante valores atípicos en los datos.

C. Clasificación de niveles de estrés

La puntuación continua se convierte en niveles discretos 1 a 5 mediante umbrales equidistantes (1.5, 2.5, 3.5, 4.5). El código recorre cada umbral y asigna el nivel correspondiente (ver función `classify_scores`). Esta discretización facilita la interpretación y acción (por ejemplo, diferenciar estudiantes “moderadamente” de “muy” estresados). Los valores de umbral se fijaron de forma que cada nivel abarque un rango unitario de la puntuación original (tras el sesgo inicial), simplificando la categorización sin ajuste estadístico complejo.

D. Cálculo de causas de estrés

Para identificar causas principales, se agrupan los factores de riesgo en tres dominios (económico, motivacional, emocional) y se calculan sub-puntuaciones ponderadas normalizadas. Por ejemplo, la puntuación económica se define como:

$$C_{\text{econ}} = \frac{\sum_{i \in I_{\text{econ}}} w_i R_i}{\sum_{i \in I_{\text{econ}}} w_i} \tag{3}$$

donde I_{econ} son los factores económicos. Análogamente para C_{mot}. Luego se define $C_{\text{emo}} = 1 - \max(C_{\text{econ}}, C_{\text{mot}})$ para completar la interpretación residual. Finalmente, se etiqueta "económica" o "motivacional" si la puntuación respectiva excede el umbral (0.3), o "emocional" si ninguna de las anteriores califica. Este diseño asegura que sólo causas con suficiente evidencia influyan en la etiqueta, evitando asignaciones triviales y permitiendo múltiples causas si aplican.

IV. Modelos matemáticos de los componentes

A. Normalización Z-score

Como se describió, cada variable continua X se estandariza restando su media μ y dividiendo por la desviación típica σ, según:

$$Z = \frac{X - \mu}{\sigma} \tag{4}$$

Este procedimiento lleva los datos a media cero y varianza uno, facilitando la combinación ponderada de variables con escalas diferentes. En el código, esto se implementa directamente con el cálculo $(X - \mu)/\sigma$.

B. Combinación ponderada de factores de riesgo

Sea $R_{i,j}$ el valor del factor de riesgo i para el estudiante j. Se define la puntuación de estrés cruda como una combinación lineal:

$$\text{score\_crudo}_j = \text{BIAS} + \sum_i w_i R_{i,j} \tag{5}$$

donde w_i son los pesos heurísticos. El sesgo (BIAS = 1.5) asegura que todos los puntajes tengan un nivel base. Para mantener la puntuación dentro del rango final 1–5 se aplica saturación:

$$\text{STRESS\_SCORE}_j = \min(\text{score\_crudo}_j, 4{,}0) + 1{,}0 \tag{6}$$

Así, cualquier score_crudo superior a 4 produce el nivel máximo 5. En el código `compute_stress_scores` se realiza exactamente esta operación.

C. Clasificación de niveles de estrés

La puntuación continua S_j resultante se clasifica en niveles $L_j \in \{1, \ldots, 5\}$ mediante umbrales $\{1{,}5, 2{,}5, 3{,}5, 4{,}5\}$. Es decir:

$$L_j = \begin{cases} 1, & S_j \leq 1{,}5, \\ 2, & 1{,}5 < S_j \leq 2{,}5, \\ 3, & 2{,}5 < S_j \leq 3{,}5, \\ 4, & 3{,}5 < S_j \leq 4{,}5, \\ 5, & S_j > 4{,}5. \end{cases} \tag{7}$$

Esta regla por umbrales se implementa iterativamente en el código, asignando el menor nivel cuyo umbral no se excede.

D. Clasificación de causas por puntuaciones parciales

Para cada categoría de causa (económica, motivacional, emocional) se calcula una puntuación normalizada. Por ejemplo, si I_{econ} es el conjunto de factores económicos, entonces la puntuación económica se define como:

$$C_j^{\text{econ}} = \frac{\sum_{i \in I_{\text{econ}}} w_i R_{i,j}}{\sum_{i \in I_{\text{econ}}} w_i} \tag{8}$$

Análogamente C_j^{mot} para los factores motivacionales. La parte emocional se considera como:

$$C_j^{\text{emo}} = 1 - \max(C_j^{\text{econ}}, C_j^{\text{mot}}) \tag{9}$$

Si la puntuación cruda de estrés S_j del estudiante es al menos 3.5, se etiqueta “económica” si $C_j^{\text{econ}} \geq 0{,}3$, “motivacional” si $C_j^{\text{mot}} \geq 0{,}3$, y “emocional” en caso de que las anteriores no apliquen y $C_j^{\text{emo}} \geq 0{,}3$. Esta lógica reproduce las funciones `compute_cause_scores` y `assign_cause` del código.

Referencias

[1] pandas Documentation. `https://pandas.pydata.org/`

[2] Google AI Gemini. `https://ai.google.dev/`

[3] openpyxl Documentation. `https://openpyxl.readthedocs.io/`