



Gabriel García Álvarez
Adrián Martín Malmierca
Victoria Pérez Lasso
Claudia Natacha Solís Costa
Hilario Javier Del Valle Escolar

ÍNDICE

Introducción

- Justificación del Proyecto
- Objetivo de la Aplicación
- Teorías de Estilos de Aprendizaje
 1. Teoría de los Estilos de Aprendizaje de David Kolb
 2. Teoría de las Inteligencias Múltiples de Howard Gardner
 3. Teoría del Desarrollo Cognitivo de Jean Piaget
 4. Teoría Sociocultural del Aprendizaje de Lev Vygotsky
 5. Teoría de la Motivación de Autodeterminación de Deci y Ryan
 6. Teoría del Locus de Control de Julian Rotter
 7. Teoría de la Autoeficacia de Albert Bandura
 8. Teoría de los Estilos Cognitivos

Estado del Arte

- Plataformas y Modelos Predictivos de Rendimiento Académico
- Lo que diferencia este proyecto de otras alternativas

Desarrollo de la idea Propuesta

Explicación de la Interfaz

- Inicio de Sesión y Registro
- Formulario de Evaluación del Alumno
- Sugerencias Basadas en IA

Explicación del Código

- Librerías y Lectura del Archivo CSV
- Limpieza de Datos y Preprocesamiento
- Entrenamiento del Modelo con SMOTE y Random Forest
- Interpretación de Resultados
- Generación de Sugerencias Personalizadas con SHAP

Resultados Obtenidos

- Error Cuadrático Medio (MSE)
- Coeficiente de Determinación (R^2)
- Interpretación de Resultados

Conclusión

- Rendimiento del Modelo
- Posibles Mejoras y Futuras Implementaciones

INTRODUCCIÓN

La idea de este proyecto ha surgido de la necesidad de apoyo académico de muchos estudiantes y de su falta de recursos. Muchas veces el alumno se encuentra con ciertas dificultades que no es capaz de comprender y/o solucionar, ahí es cuando entra en juego nuestra app.

El problema es interesante porque aborda una de las principales dificultades en el ámbito educativo: la falta de personalización en el apoyo académico. Cada estudiante tiene un estilo de aprendizaje, nivel de motivación y circunstancias individuales que afectan su desempeño, pero los sistemas educativos tradicionales tienden a ofrecer soluciones genéricas que no se adaptan a estas particularidades.

Nuestra aplicación es capaz de predecir mediante inteligencia artificial la nota de un alumno, basándose en un conjunto de características relacionadas con su desempeño académico y hábitos de estudio.

También es capaz de proporcionar sugerencias de áreas en las que el alumno podría mejorar al comparar sus características con las de otros estudiantes con mejores notas.

Esto lo consigue gracias a que el modelo utiliza un algoritmo de regresión Random Forest. Este modelo ha sido previamente entrenado con una amplia cantidad de datos sobre hábitos estudiantiles y rendimiento académico.

Esta aplicación está creada de manera sencilla e intuitiva, para así facilitar el uso de la misma a una amplia gama de usuarios de diferentes edades y países.

Hay diferentes tipos de estudiantes en función a su forma de pensar y a cómo afrontan diferentes situaciones. Hay diferentes tipos en función de las diferentes teorías existentes.

1. Teoría de los Estilos de Aprendizaje de David Kolb

David Kolb dijo que cada persona aprende de diferente manera, donde el aprendizaje es un proceso cíclico que consta de cuatro etapas. Los estudiantes se pueden clasificar en diferentes tipos, en función de su preferencia por las forma de aprender:

- **Convergente:** Aplican ideas a situaciones prácticas. Se preocupa en resolver problemas.
- **Divergente:** Observan e imaginan diversas soluciones. Además, tienen gran creatividad.
- **Asimilador:** Entienden conceptos y teorías abstractas. Son más teóricos.
- **Acomodador:** Aprenden mejor a través de la acción y la experiencia directa. Se declinan la práctica sobre la teoría.

2. Teoría de las Inteligencias Múltiples de Howard Gardner

Gardner expuso que hay diferentes tipos de inteligencia y que cada persona tiene una combinación de ellas, la cual es única. Los estudiantes se pueden clasificar según su tipo predominante de inteligencia:

- **Inteligencia lingüística:** Aprenden mejor a través del lenguaje, escritura y lecturas.
- **Inteligencia lógico-matemática:** Prefieren el pensamiento abstracto, el razonamiento y la lógica.
- **Inteligencia espacial:** Prefieren las imágenes, visualizaciones y el espacio.
- **Inteligencia musical:** Aprenden mejor con sonidos, música y ritmos.
- **Inteligencia corporal-cinestésica:** Usan el cuerpo para expresarse o aprender, prefieren el movimiento.
- **Inteligencia interpersonal:** Aprenden con la interacción social.
- **Inteligencia intrapersonal:** Son hábiles entendiendo sus propios pensamientos y sentimientos.
- **Inteligencia naturalista:** Se relacionan mejor con el entorno natural.

3. Teoría del Desarrollo Cognitivo de Jean Piaget

Piaget expuso cómo los niños y adolescentes pasan por diferentes etapas de desarrollo cognitivo, lo que también afecta el tipo de estudiante que son:

- **Etapas sensoriomotora (0-2 años):** Aprenden a través de la interacción física directa con el entorno.
- **Etapas preoperacional (2-7 años):** Piensan en términos de imágenes y símbolos, pero no usan aún la lógica formal.
- **Etapas de las operaciones concretas (7-11 años):** Pueden pensar lógicamente, pero sólo sobre objetos y situaciones concretas.
- **Etapas de las operaciones formales (11 años en adelante):** Pueden pensar de forma lógica, abstracta y realizar razonamientos hipotéticos.

4. Teoría Sociocultural del Aprendizaje de Lev Vygotsky

Vygotsky expuso la importancia del contexto cultural y social en el aprendizaje. Los estudiantes aprenden mejor cuando están en su zona de desarrollo próximo, es decir, cuando se encuentran ante un desafío que no pueden resolver solos, pero sí con la ayuda de otros, ya sean compañeros o profesores. Los tipos de estudiantes en esta teoría son:

- **Estudiantes autónomos:** Pueden aprender solos o con mínima ayuda, por lo que están en el límite superior de la zona de desarrollo próximo.
- **Estudiantes dependientes:** Necesitan más apoyo y guía externa para resolver problemas.

5. Teoría de la Motivación de Autodeterminación de Deci y Ryan

Distingue diferentes tipos de motivación en los estudiantes:

- **Estudiantes con motivación intrínseca:** Aprenden porque disfrutan el proceso de aprender en sí mismo. Suelen ser más persistentes y curiosos.
- **Estudiantes con motivación extrínseca:** Aprenden para obtener recompensas externas, como reconocimiento, calificaciones o evitar castigos.
- **Estudiantes desmotivados:** No encuentran razones internas ni externas para aprender, por lo que suelen tener bajo rendimiento y baja satisfacción.

6. Teoría del Locus de Control de Julian Rotter

El locus de control se refiere a la creencia de una persona sobre si puede controlar los eventos que afectan su vida. En el contexto educativo, esto afecta el tipo de estudiante:

- **Estudiantes con locus de control interno:** Piensan que sus esfuerzos determinan su éxito o fracaso. Suelen ser más responsables y proactivos.
- **Estudiantes con locus de control externo:** Piensan que su éxito o fracaso depende de factores externos, como acciones de otros, la suerte... Pueden ser más reactivos o pasivos.

7. Teoría de la Autoeficacia de Albert Bandura

La autoeficacia es la creencia que tiene una persona en su capacidad de realizar una tarea específica. Los estudiantes pueden clasificarse como:

- **Estudiantes con alta autoeficacia:** Tienen confianza en su capacidad para resolver problemas y aprender. Suelen tener mejores resultados debido a que son más persistentes.
- **Estudiantes con baja autoeficacia:** Dudan de sus habilidades y se desmotivan fácilmente ante dificultades.

8. Teoría de los Estilos Cognitivos

Los estilos cognitivos hacen referencia a cómo las personas procesan la información. Los estudiantes se clasifican en:

- **Estudiantes visuales:** Anteponen aprender con diagramas, gráficos y contenido visual.
- **Estudiantes auditivos:** Aprenden mejor escuchando, a través de discusiones o explicaciones orales.
- **Estudiantes kinestésicos:** Prefieren actividades prácticas, necesitan interactuar físicamente con el material de aprendizaje.

Debido a estos múltiples intentos de clasificar a los alumnos, creemos propio intentar ayudarles a mejorar y el hecho de que con una mayor cantidad de datos de entreno, también mejoraría considerablemente el rendimiento del modelo.

ESTADO DEL ARTE

En el estado del arte sobre predicción de rendimiento académico y personalización educativa, existen herramientas y estudios previos que han explorado enfoques similares al proyecto presentado. Destacamos algunas alternativas y enfoques relevantes en la literatura:

Plataformas y Modelos Predictivos de Rendimiento Académico

Existen herramientas diseñadas para predecir el desempeño de los estudiantes, aunque estas no siempre incluyen explicabilidad o recomendaciones personalizadas:

- **EdTech Platforms (Coursera, Khan Academy):** estas plataformas utilizan algoritmos de machine learning para personalizar el contenido según el progreso del estudiante, pero se centran más en recomendaciones de materiales que en predicciones de notas.
- **Sistemas de gestión del aprendizaje (LMS, como Moodle):** integran modelos básicos para monitorear el progreso y rendimiento, pero no suelen generar explicaciones detalladas ni sugerencias específicas.
- **Enfoques predictivos basados en regresión:** estudios como los de Marbouti et al. (2016) utilizan modelos como regresión logística para predecir el éxito académico. Sin embargo, estos modelos a menudo carecen de interpretabilidad y se limitan a predicciones sin un análisis profundo de los factores.

Lo que diferencia este proyecto de otras alternativas:

1. **Integración de Predicción y Recomendación:** A diferencia de herramientas existentes que solo monitorean o predicen, este proyecto utiliza los resultados del modelo para generar recomendaciones personalizadas basadas en datos explicables (SHAP).
2. **Explicabilidad del Modelo:** Herramientas como SHAP permiten entender qué factores afectan las predicciones, haciéndolas más transparentes y útiles para la toma de decisiones educativas.
3. **Diseño centrado en el usuario:** La aplicación está diseñada tanto para estudiantes como para profesores, lo que fomenta una interacción más colaborativa.

DESARROLLO DE LA IDEA PROPUESTA

Explicación de la Interfaz

Interfaz del alumno

En la Figura 1 podemos ver la página de inicio de la aplicación. Esta da la opción de iniciar sesión para poder hacer uso de la app o acceder a la página de registro (Figura 2) si no lo has hecho previamente. Depende de quién se registre o inicie sesión (alumno o profesor), la interfaz se mostrará de una manera u otra.

Figura 1.

The figure consists of two screenshots of the EDUCAREAI application interface, presented on a dark background.

Top Screenshot (Login Page):

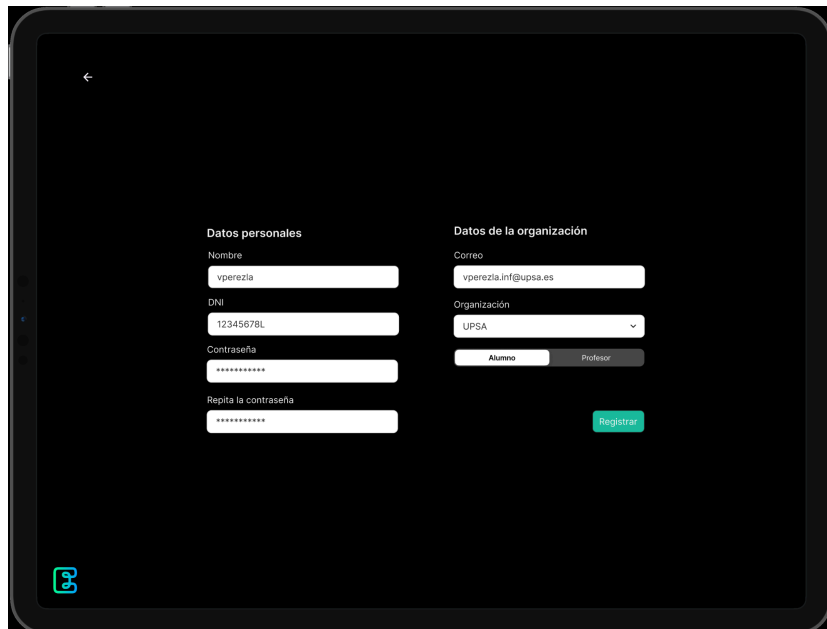
- Logo: A stylized 'E' icon above the text 'EDUCAREAI' and the tagline 'REFUERZA HOY. TRIUNFA MAÑANA'.
- Buttons: A green 'Registrarse' button in the top right corner and a green 'Iniciar sesión' button below the password field.
- Form Fields: Two white input fields labeled 'Nombre' (containing 'Victoria') and 'Contraseña' (containing masked characters).

Bottom Screenshot (Registration Page):

- Navigation: A back arrow icon in the top left corner.
- Form Structure: Two columns of form fields. The left column is titled 'Datos personales' and the right column is titled 'Datos de la organización'.
- 'Datos personales' Fields: 'Nombre' (Victoria), 'DNI' (12345678L), 'Contraseña' (masked), and 'Repita la contraseña' (masked).
- 'Datos de la organización' Fields: 'Correo' (vperezia.inf@upsa.es), 'Organización' (a dropdown menu showing 'UPSA'), and a role selector with 'Alumno' and 'Profesor' options.
- Buttons: A green 'Registrar' button at the bottom right.
- Footer: A small version of the EDUCAREAI logo in the bottom left corner.

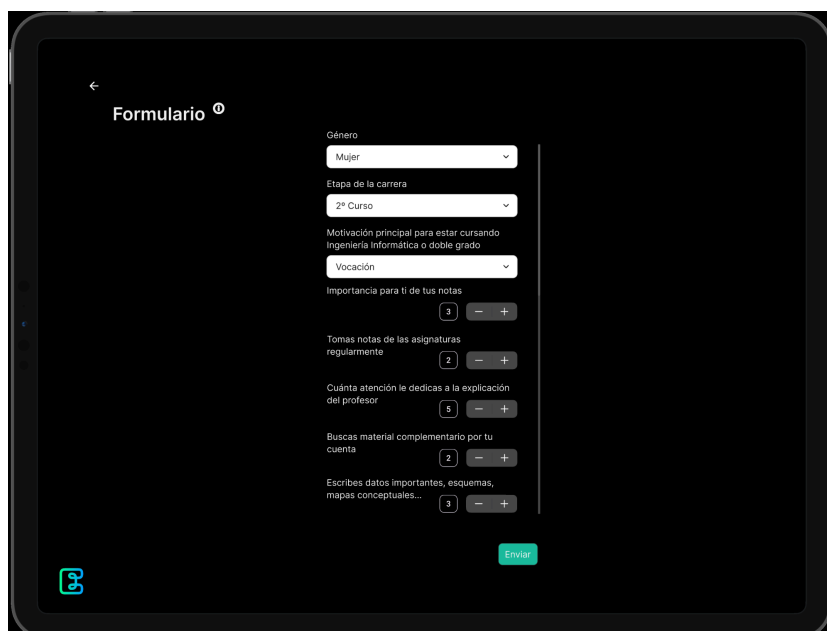
Al iniciar la aplicación, se mostrará una pantalla de inicio de sesión donde el usuario deberá ingresar su nombre de usuario y contraseña. Si el usuario no está registrado, tendrá la opción de hacer clic en el botón "Registrarse", lo que lo llevará a la página de registro. Una vez completado el registro, podrá acceder a la aplicación y utilizar sus funcionalidades.

Figura 2.



Una vez registrado o tras haber iniciado sesión, en el caso del alumno, se redirigirá a un formulario (Figura 3) donde se te harán varias preguntas sobre el alumno y sus hábitos de estudio para conocer y poder evaluarlo posteriormente.

Figura 3



Una vez dentro de la aplicación, el usuario verá un formulario que, si ha iniciado sesión previamente, aparecerá pre rellenado con los datos de su última sesión, permitiendo modificarlos si lo desea. En caso de ser su primera vez en la plataforma, el formulario estará vacío, listo para ser completado. Los datos ingresados en este formulario son las variables que se utilizarán para entrenar la red neuronal y realizar las predicciones personalizadas.

Figura 4

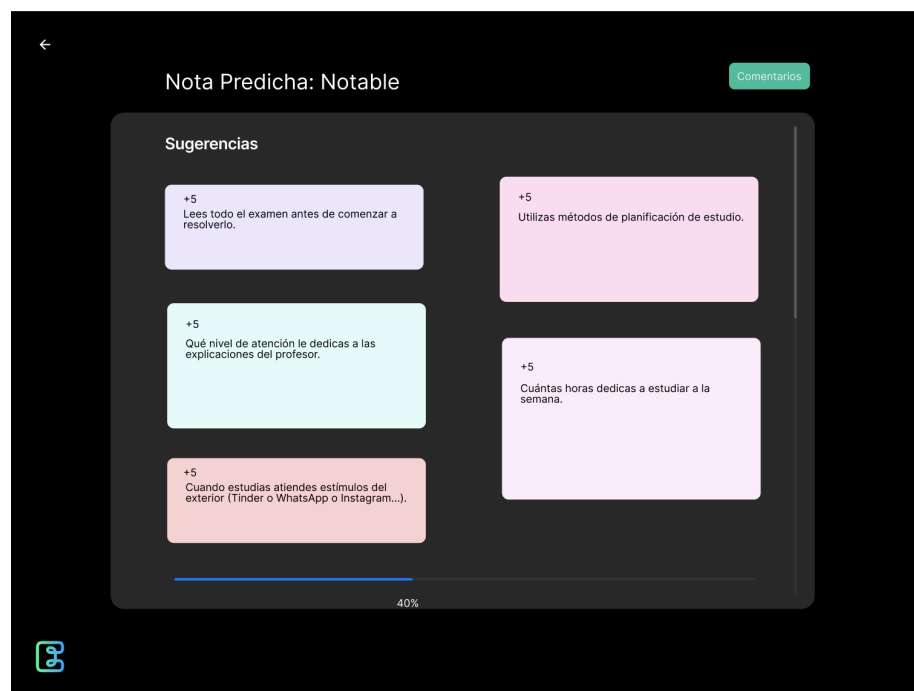


Figura 5.



Después de enviar los datos del formulario, el alumno será redirigido a la sección de tareas (Figura 4). En esta área, nuestro modelo de IA proporcionará sugerencias y recomendaciones personalizadas para mejorar su rendimiento académico. En la parte superior derecha, habrá un botón que permitirá acceder a los comentarios añadidos por el profesor, así como a contenido multimedia adicional (figura 5).

Interfaz del profesor

Figura 6.

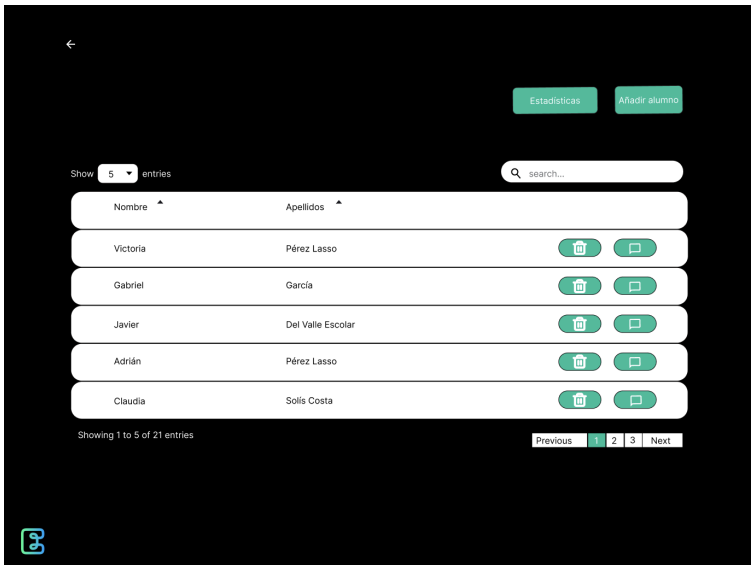


Figura 7.

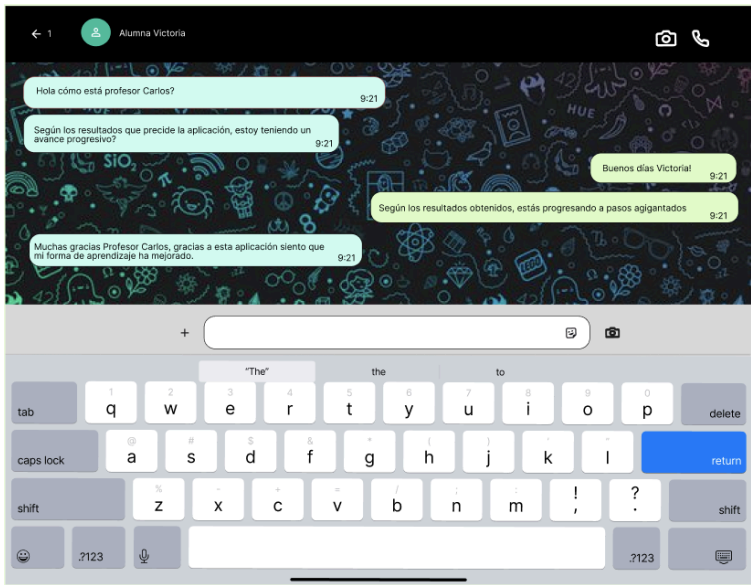


Figura 8.

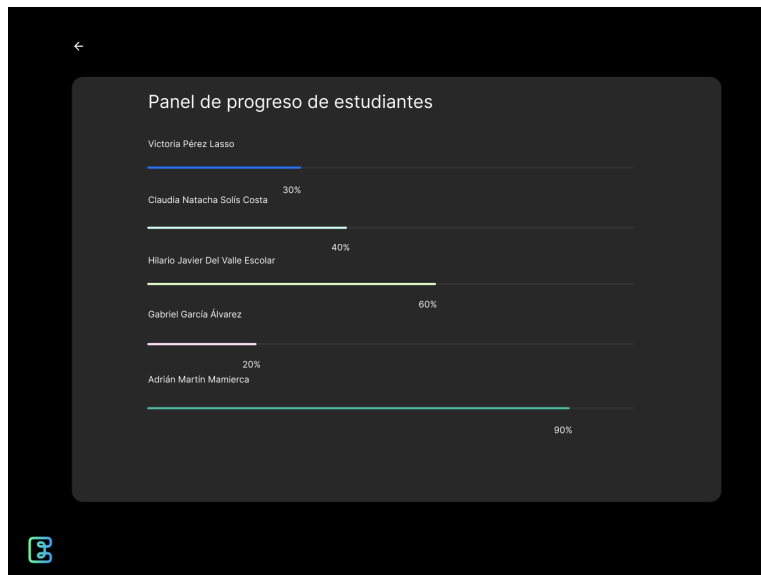


Figura 9.

Formulario para agregar un nuevo alumno:

Nombre alumno:

DNI alumno:

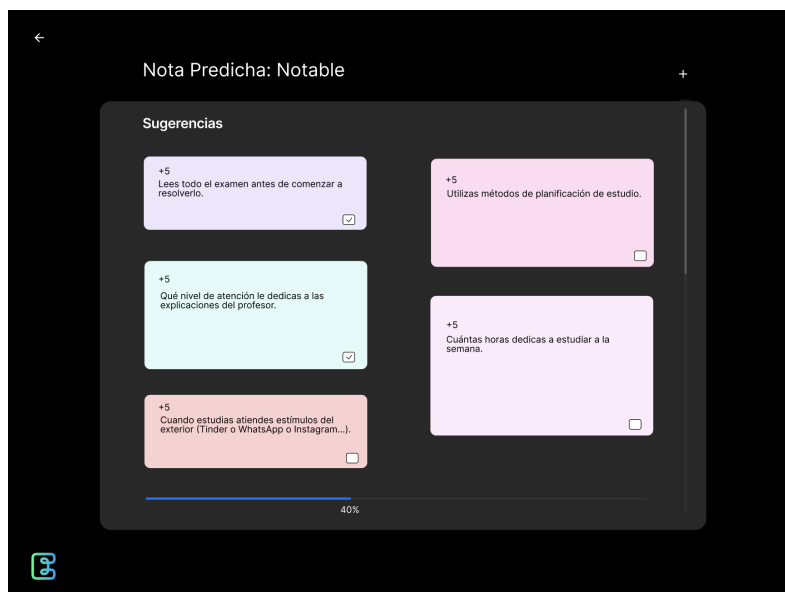
The figure shows a dark-themed interface with a form for adding a new student. It includes two input fields for 'Nombre alumno' and 'DNI alumno', and a button labeled 'Agregar'. A small icon is visible in the bottom left corner.

Una vez que el profesor acceda a la aplicación, verá una tabla que incluye todos sus alumnos. Desde esta vista, tendrá la opción de eliminar usuarios o añadir nuevos, así como iniciar conversaciones con ellos.

Si desea agregar un alumno a su lista, simplemente debe hacer clic en el botón ubicado en la parte superior derecha de la pantalla.

Para consultar los avances de todos los alumnos, puede dirigirse al botón etiquetado como “Estadísticas”.

Figura 9.



Si el profesor presiona sobre cualquier celda de la tabla en la que aparecen los estudiantes, accede a las sugerencias que le haya generado el modelo, si es que alguna vez lo ha rellenado. Además, mostrará una barra con el progreso de las tareas que haya completado, es decir, si ha conseguido mejorar en alguno de estos aspectos. Si ha mejorado, eso lo comprueba el profesor y marca una de las sugerencias como completadas; solo pueden ser modificadas por el profesor. El botón “+” nos llevará a la siguiente página.

Figura 10.



Esta es la página a la que accederemos, en la que se puede ver el contenido multimedia para el alumno y los comentarios adyacentes que haya puesto el propio profesor y otros

profesores. El profesor tendrá permisos para añadir material a las dos columnas del estudiante.

Explicación del código

Cargamos las librerías que necesitamos y leemos el archivo csv mediante la librería pandas.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from imblearn.over_sampling import SMOTE
from sklearn.impute import SimpleImputer
import shap
import numpy as np

# Leemos el csv
data = pd.read_csv('DATOSJAKATON.csv')
```

El siguiente paso natural tras leer los datos será limpiarlos. Para esto, convertimos la variable que vamos a predecir (nota media) en numérica haciendo un mapping. Luego, pasamos las variables categóricas a binarias mediante One-Hot Encoding gracias a la función de pandas `get_dummies()`.

Este método construye una columna binaria por cada valor que pueda tomar una variable. Por ejemplo, la variable Motivación principal contiene varios valores: Prestigio, Remuneración económica, Vocación... Motivación principal se transformará de manera que nos quedará una variable por cada valor que pueda tomar, y valdrá 0 o 1.

```
# Convertimos variables categóricas a numéricas mediante mapping o One-Hot Encoding
nota_mapping = {'Suspense': 0, 'Aprobado': 1, 'Notable': 2, 'Sobresaliente': 3, 'Matrícula de Honor': 4}
data['nota_num'] = data['Nota media en el curso'].map(nota_mapping)

data_clean = data.drop(columns=['Marca temporal', 'Nota media en el curso', 'nota_num'])
data_encoded = pd.get_dummies(data_clean, drop_first=True)

# Rellenamos los NaN que hayan quedado
imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(data_encoded)
```

Antes de entrenar el modelo, dividimos entre variables a usar y variables objetivo, para después aplicar SMOTE.

SMOTE (Synthetic Minority Oversampling TEchnique) sirve para eliminar el sesgo de algoritmos basados, por ejemplo, en árboles de decisión. Esta técnica crea muestras sintéticas que establecen un equilibrio entre las muestras; en nuestro caso, entre aprobados y suspensos.

```
# Recogemos los datos que vamos a usar en el modelo, separando las variables y la nota que queremos predecir
X = pd.DataFrame(X_imputed, columns=data_encoded.columns)
y = data['nota_num']

# Aplicamos SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)
```

Una vez todo queda dispuesto, entrenamos el modelo de Random Forest con el 70% de los datos, dejando un 30% para hacer el testing.

Random Forest es un algoritmo de machine learning basado en árboles de decisión. Un árbol de decisión es un esquema, el cual plantea una pregunta. Mediante otras preguntas nos ayudará a alcanzar una hoja o nodo final, que contendrá una de las respuestas binarias a la pregunta inicial. Random Forest combina múltiples árboles de decisión para encajar a las nuevas muestras.

```
# Dividimos en entrenamiento y prueba para entrenar al Random Forest
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.3, random_state=42)

model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)
```

La siguiente función simplemente nos ayuda a mapear la nota que obtengamos del modelo de vuelta a una más fácilmente interpretable por el usuario.

```
def interpretar_nota(nota_predicha):
    if nota_predicha < 0.5:
        return "Suspenso"
    elif nota_predicha < 1.5:
        return "Aprobado"
    elif nota_predicha < 2.5:
        return "Notable"
    elif nota_predicha < 3.5:
        return "Sobresaliente"
    else:
        return "Matrícula de Honor"
```

La siguiente función será la que emplee el código anterior para dar un resultado concreto al estudiante.

Primero, usa la función predict del model para darnos un resultado y lo interpreta.

Luego, crearemos las sugerencias. Estas sugerencias irán en base a valores SHAP, una herramienta extremadamente útil a la hora de darle una explicabilidad al modelo, haciéndolo más transparente. SHAP (SHapley Additive exPlanations) es un planteamiento para explicar los resultados de un machine learning basado en la teoría de juegos.

Nosotros valoraremos la importancia que le dé SHAP a las variables junto con el valor que tenga el alumno. Se dará un valor SHAP negativo a aquellas variables con un impacto negativo y de valor alto para el estudiante, y aquellas de impacto positivo y de valor bajo para el estudiante. Luego, ordenaremos por las más negativas. Así podremos crear las sugerencias personalizadas, hasta un máximo de cinco sugerencias.

```
def predecir_nota_alumno_shap(caracteristicas_alumno):
    alumno_df = pd.DataFrame([caracteristicas_alumno], columns=X.columns)

    nota_predicha = model.predict(alumno_df)[0]
    nota_interpretada = interpretar_nota(nota_predicha)

    print(f"La nota predicha para el alumno es: {nota_predicha:.2f} ({nota_interpretada})")

    # Generamos los valores SHAP para este estudiante en concreto
    explainer = shap.TreeExplainer(model)
    shap_values = explainer.shap_values(alumno_df)

    # Guardamos juntos el nombre de la variable, el valor del alumno para ella y el valor SHAP que le asociaremos
    shap_df = pd.DataFrame({
        'Característica': X.columns,
        'Valor actual': alumno_df.iloc[0],
        'Valor SHAP': shap_values[0]
    })

    # Cogemos solo las características con valores SHAP negativos, es decir, que reducen la nota
    shap_negativo = shap_df[(shap_df['Valor SHAP'] < 0) & (~shap_df['Valor actual'].isna())]

    # Ordenamos los valores SHAP por los más negativos y cogemos solo los primeros 5
    shap_negativo = shap_negativo.sort_values(by='Valor SHAP')
    sugerencias = shap_negativo.head(5)

    if not sugerencias.empty:
        print("\nSugerencias de mejora basadas en SHAP (máximo 5):")
        for _, row in sugerencias.iterrows():
            print(f"- {row['Característica']}: Valor actual del alumno -> {row['Valor actual']}.")
    else:
        print("\nNo hay sugerencias de mejora, todas las características tienen un impacto positivo o no se encontraron valores significativos.")
```

RESULTADOS OBTENIDOS

Actualmente, tenemos un MSE o error cuadrático medio de 0.1323 y un MAE o error absoluto medio de 0.2648. Este primer error representa la diferencia entre lo predicho y los valores reales al cuadrado. El segundo error representa la media de las diferencias absolutas entre los valores reales y las predicciones.

En cuanto a rendimiento, lo podemos medir con el Coeficiente de Determinación, que mide cómo de bien se ajustan las predicciones al resultado real o, dicho de otro modo, qué proporción de la variabilidad en la variable objetivo puede explicarse por las características del modelo. No podemos hablar en un modelo de este tipo directamente de precisión, porque estamos trabajando con una variable predicha continua, no en un algoritmo de clasificación.

En nuestro caso, hemos obtenido un R^2 de 0,89: el modelo explica el 89% de la variabilidad en las notas. Significa que el modelo es bueno para explicar la relación entre las características predictoras y las notas.

CONCLUSIÓN

Gracias a esta aplicación podemos proporcionar apoyo escolar de una manera universal, sencilla y adaptable. Los dos principales puntos fuertes de esta aplicación serán:

1. Predicción precisa de las notas de los estudiantes

El modelo Random Forest entrenado es capaz de predecir las notas de los estudiantes con un alto grado de rendimiento.

2. Sugerencias personalizadas basadas en explicaciones interpretables

El uso de SHAP ha permitido que el modelo no solo prediga las notas, sino también proporciona sugerencias claras y útiles sobre cómo los estudiantes pueden mejorar en áreas clave para aumentar sus resultados. Esto hace que el modelo sea más útil para los usuarios finales (en este caso, los estudiantes y profesores) al ofrecer recomendaciones prácticas basadas en datos reales.

El proyecto combina tanto técnicas de machine learning como enfoques de interpretabilidad para no solo predecir, sino también explicar y mejorar el rendimiento de los estudiantes, lo que tiene un gran potencial para aplicaciones educativas.

Perspectivas futuras

- Más datos para mejorar la precisión: Se podrían agregar más características sobre el comportamiento del estudiante o expandir el conjunto de datos para mejorar aún más el rendimiento del modelo.
- Aplicación práctica: Este modelo se podría integrar, además de como una aplicación propia, en otras plataformas educativas.

ANEXO

El vídeo se encuentra disponible dentro de Github

Github: <https://github.com/Gab277/EduCareAI>