TECNICATURA SUPERIOR EN

Ciencia de Datos e Inteligencia Artificial

PRACTICA PROFESIONALIZANTE II

TEC. SUPERIOR EN CIENCIAS DE DATOS E INTELIGENCIA ARTIFICIAL - ISPC

EA 3

INTEGRANTES:

BLASICHE Andrés, DNI 31405881

CABRERA Marcos Rodrigo, DNI 31667009

PALOMEQUE Dalila Macarena, DNI 39733230

TEJEDA Romina Soledad, DNI 34501801

https://trello.com/b/uGAiwKkv/practica-profesionalizante-ii

https://github.com/ISPC-23/ABP_practica_prof_2

https://colab.research.google.com/drive/1EqyfOHUwmvIfLBRGGRpOmrRYG-bQ0epV?usp=sharing

ÍNDICE

NOMBRE DEL PROYECTO

Predicción Justa del Rendimiento Académico Estudiantil

(¿Qué?) Predicción del rendimiento académico

(¿Cómo?) Mediante técnicas de regresión con enfoque en la equidad

(¿Cuándo?) Durante mayo y junio de 2025

TIPO DE PROYECTO

Tecnológico, con enfoque ético y educativo

Desarrollo de un modelo de machine learning con enfoque en equidad.

ESPACIO CURRICULAR O ESPACIOS PARTICIPANTES EN EL MÓDULO

Práctica Profesionalizante II de la Tecnicatura Superior en Ciencias de Datos e Inteligencia Artificial

EJES TEMÁTICOS / RED DE CONCEPTOS

Metodologías y Ciclo de Vida de Proyectos

Herramientas de gestión y colaboración - Mlflow

Ética, Gobernanza y Despliegue - Fairlearn: Equidad en Modelos de IA

PROBLEMÁTICA / NECESIDAD

En el entorno escolar, los estudiantes enfrentan múltiples factores que afectan su rendimiento académico. Sin embargo, las notas no reflejan la totalidad de estos factores. Existe una necesidad de **herramientas predictivas justas y explicables** que permitan comprender cómo influyen los hábitos diarios en el desempeño, sin generar discriminación o reforzar estereotipos.

FUNDAMENTACIÓN

Elegimos esta problemática porque representa un desafío real en el ámbito educativo: medir con precisión, sin injusticias, el rendimiento estudiantil. El potencial del proyecto es alto, ya que podría convertirse en la base de herramientas de orientación personalizadas. Está directamente relacionado con el perfil profesional del científico de datos responsable, ético y comprometido con el impacto social. Además, puede tener un **impacto positivo en la comunidad educativa**, aportando información útil para estudiantes, docentes y familias.

VISIÓN DEL PROYECTO Y OBJETIVOS

Objetivo General:

Desarrollar un modelo de regresión para estimar el rendimiento académico de estudiantes en base a sus hábitos diarios, incorporando métricas de equidad que aseguren predicciones justas según género

Objetivos Específicos:

1. Analizar el dataset Student Habits vs Academic Performance con EDA y Fairlearn.
2. Preparar los datos con enfoque ético: limpieza, codificación, mitigación de sesgos.
3. Desarrollar y registrar modelos predictivos usando MLflow.
4. Evaluar el modelo con métricas técnicas y de equidad.
5. Generar recomendaciones generales para mejorar hábitos estudiantiles.

SELECCIÓN DE ACCIONES

Objetivo	Acción
1. Analizar el dataset con EDA y Fairlearn	- Exploración de variables - Visualización de distribución y correlaciones
2. Preparar datos para modelado	- Limpieza de valores nulos - Codificación de variables categóricas - Mitigación de sesgos
3. Desarrollar modelo con MLflow	- Entrenar modelos de regresión - Registrar métricas y parámetros - Comparar versiones
4. Evaluar el modelo	- Métricas R^2, MAE, RMSE - Métricas de equidad con Fair Learn - Validación cruzada
5. Generar recomendaciones	- Interpretación de variables significativas - Elaboración de sugerencias generales

CRONOGRAMA

Etapa	Duración Estimada	Fechas
Análisis Exploratorio (EDA) y Fair Learn	2 semanas	del 1 al 15 de mayo
Preparación de Datos (limpieza, codificación, mitigación de sesgos, feature engineering)	2 semanas	del 16 al 30 de mayo
Modelado Inicial y Experimentación con MI flow	2 semanas	del 1 al 15 de junio
Despliegue del Modelo en	2 semanas	del 16 al 30 de junio

Producción (API / App Web con Mlflow)		

PRODUCTO FINAL

El producto final será un modelo predictivo del rendimiento académico de estudiantes, basado en sus hábitos diarios y evaluado con métricas de precisión y equidad. A partir de sus resultados, se generará un informe visual con recomendaciones generales para mejorar el desempeño escolar, dirigido a estudiantes, docentes y familias. Este material podrá compartirse en redes sociales o espacios educativos, y servirá como base para el desarrollo futuro de una app de recomendaciones personalizadas.

Descripciones

Descripción del dataset

Variables:

- **age**: Edad del estudiante.(Int.)
- **gender**: Maculino/Femenino/Otro.(Object.)
- **study_hours_per_day**: Promedio de horas de estudio por dia.(Float.)
- **social_media_hours**: Tiempo que los estudiantes pasan usando redes sociales (Hs).(Float)
- **netflix_hours**: Tiempo promedio que pasan diariamente viendo Netflix.(Float.)
- **part_time_job**: Se refiera a si el estudiante tiene trabajo de media jornada.(Object)
- **attendance_percentage**: Porcentaje de asistencia a clases.(Float.)
- **sleep_hours**: Promedio de horas de sueño por estudiante.(Float.)
- **diet_quality**: Calidad de la dieta, baja, media, buena.(Object.)
- **exercise_frequency**: Cantidad de veces por semana que hace ejercicio.(Int.)
- **parental_education_level**: Nivel de educación de los padres. Secundario, universitario, otro.(Object.)
- **internet_quality**: Calidad de internet. Bueno, promedio, otro.(Object.)

- **mental_health_rating**: Calificación de salud mental, en escala de 1-10(Int.)
- **extracurricular_participation**: Realiza actividades extracurriculares fuera del colegio.(Object.)
- **exam_score**: Puntuación o nota en el examen final (0-100)(Float.)

Métricas clave a evaluar

- MSE
- R^2
- Fairness

Exploración y Preprocesamiento de Datos

1. Limpieza de datos:
   La **limpieza de datos** (o *data cleaning*) es una etapa fundamental en cualquier proyecto de ciencia de datos o *machine learning*. Consiste en identificar y corregir (o eliminar) errores, inconsistencias, o valores no deseados en un conjunto de datos para mejorar su calidad y asegurar que los análisis y modelos que se construyan sean confiables. En nuestro caso de estudio encontramos solo valores nulos (Nan) los cuales se encontraban en la variable **parental_education_level** que al tratarse de una variable con distribución bimodal estos fueron imputados aleatoriamente con la proporción de cada una de las modas.

2. Análisis exploratorio (EDA):
   El Análisis Exploratorio de Datos es una de las etapas más importantes al comenzar un proyecto de ciencia de datos o *machine learning*. Consiste en explorar y visualizar los datos para entender sus características, detectar patrones, anomalías, relaciones y errores antes de construir modelos.

Metodología

Para este proyecto utilizamos algoritmos de regresión de machine learning para el modelado. Se utilizó Google Collaboratory para codificar y desarrollar modelos de aprendizaje automático.

Imports:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from plotly.offline import init_notebook_mode
init_notebook_mode(connected=True)
import plotly.figure_factory as ff
import plotly.graph_objects as go
from scipy.stats import chi2_contingency
```

```
from fitter import Fitter
```

```
from scipy.stats import ttest_ind, f_oneway
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn.linear_model import Lasso, Ridge
from sklearn.model_selection import GridSearchCV
```

```
from sklearn.metrics import mean_squared_error, r2_score
```

```
from fairlearn.metrics import MetricFrame
```

```
import mlflow
```

```
from pyngrok import ngrok, conf
import getpass
```

```
from pycaret.regression import setup, compare_models, finalize_model, save_model
```

```
from flask import Flask, request, jsonify
```

```
import threading
```

```
import requests
```

Preprocesamiento de los datos

En nuestro caso, no fue necesario realizar un preprocesamiento extenso, ya que no se identificaron sesgos significativos y la cantidad de valores atípicos era muy baja. Además, al utilizar un algoritmo robusto y poco sensible a outliers, no fue prioritario abordar este aspecto en profundidad.

Nos enfocamos principalmente en la detección de valores duplicados y nulos, siendo estos últimos los más frecuentes. Para resolver este problema, analizamos particularmente la variable *parental_education_level*, que presenta una distribución bimodal. Los valores nulos en esta variable fueron imputados de manera aleatoria, respetando la proporción correspondiente a cada una de las modas.

```
df.isnull().sum()
```

	0
student_id	0
age	0
gender	0
study_hours_per_day	0
social_media_hours	0
netflix_hours	0
part_time_job	0
attendance_percentage	0
sleep_hours	0
diet_quality	0
exercise_frequency	0
parental_education_level	91
internet_quality	0
mental_health_rating	0
extracurricular_participation	0
exam_score	0

dtype: int64

Fig 1. Datos nulos o faltantes en el dataset.

```
# Extraemos la serie
educacionPadres = df['parental_education_level']
```

```
# Comprobamos cuantas veces aparece cada categoria dentro de la serie
conteo = educacionPadres.value_counts(dropna=False) # para q me muestre los Nan
print(conteo)
```

```
parental_education_level
High School    392
Bachelor       350
Master         167
NaN             91
Name: count, dtype: int64
```

```
plt.figure(figsize=(12,6))
sns.countplot(data = df, x = 'parental_education_level', order = educacionPadres.value_counts().index) #los Nan no se pueden graficar con sns
plt.title('Distribución del nivel de educacion de los padres')
plt.xticks(rotation = 45)
plt.show()
```

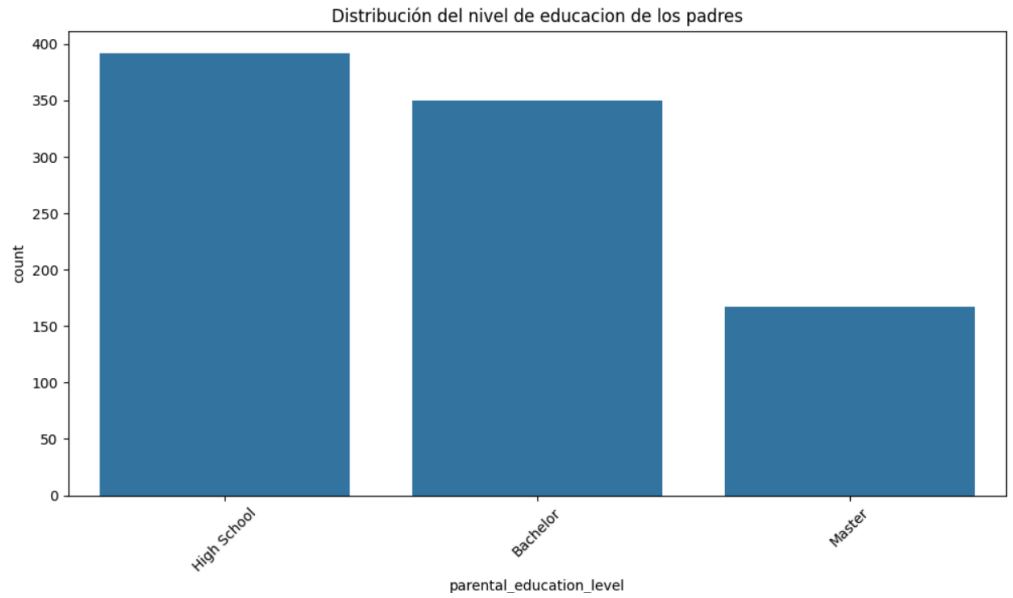

Fig 2. Tratamiento de los valores nulos.

Demostración de la bimodalidad

```
porcentajes = educacionPadres.value_counts(normalize = True,dropna=False)*100
print(porcentajes)
```

```
parental_education_level
High School    39.2
Bachelor       35.0
Master         16.7
NaN             9.1
Name: proportion, dtype: float64
```

Fig 3. Proporción de las características de la clase.

Se observa claramente una distribución bimodal en la variable, con dos categorías predominantes que representan el 74,2% de los datos. En este contexto, no es recomendable eliminar los valores nulos (NaN), ya que implicaría una pérdida significativa de información.

Si bien existe la opción de crear una nueva categoría como "desconocido", esto introduciría una clase adicional sin sustento real en los datos, lo que podría afectar la calidad del análisis.

Por ello, se optó por imputar los valores nulos respetando la distribución original de las dos modas. Específicamente, se asignaron aleatoriamente los valores faltantes a las categorías “high school” o “bachelor” en proporción a su frecuencia.

- **Ventaja**: preserva la distribución original de los datos.
- **Desventaja**: introduce un grado de aleatoriedad en el conjunto de datos.

```
# Extraemos los Nan de la serie 'parental_education_level' por su indice
indices_nan = df[df['parental_education_level'].isna()].index

# como tenemos una lista de indices que tienen los nan de ''parental_education_level''
df.loc[indices_nan,'parental_education_level'] = np.random.choice(
    ['High School','Bachelor'],
    size = len(indices_nan),
    p = [0.392/0.742, 0.35/0.742]
```

Fig 4. Imputación de los valores nulos.

Estudio de la variable respuesta: *exam_score*

Analizar la distribución de la variable respuesta es fundamental por varias razones clave:

1. Elección del modelo

   Algunos algoritmos hacen supuestos sobre la distribución de la variable respuesta. Por ejemplo:

   - Regresión lineal: se asume que los residuos (errores) del modelo siguen una distribución normal.
   - Regresión logística: requiere que la variable respuesta sea binaria o categórica.
   - Modelos probabilísticos (como Naive Bayes): la distribución de la variable afecta directamente las probabilidades y, por ende, las predicciones.

2. Transformaciones y normalización (en modelos de regresión)

   Cuando la variable respuesta presenta una distribución muy sesgada (skewed), esto puede afectar negativamente el rendimiento del modelo. En tales casos, pueden aplicarse transformaciones que buscan normalizar la distribución, tales como:

   - Transformación logarítmica: log(y)
   - Transformación de Box-Cox
   - Transformación de raíz cuadrada

Estas técnicas permiten que el modelo aprenda de manera más eficiente, sobre todo si el algoritmo se beneficia de una distribución aproximadamente normal.

3. Evaluación y validación

   Comprender la distribución de *exam_score* es útil para:

   - Mejorar la estrategia de división de datos, por ejemplo aplicando estratificación si existen clases desbalanceadas.

- Interpretar adecuadamente métricas de error (como RMSE, MAE) y su relación con distintos rangos de la variable.
- Detectar problemas de sobreajuste o falta de generalización en zonas específicas del espacio de salida.

4. Interpretabilidad

   Conocer cómo se distribuye la variable objetivo permite:

   - Identificar valores atípicos (outliers) o casos anómalos.
   - Reconocer patrones inusuales que podrían requerir un tratamiento especial.

Ajustar las expectativas sobre el comportamiento del modelo. Por ejemplo, si la mayoría de los valores están entre 10 y 20, pero existen algunos casos extremos cercanos a 1000, podría ser necesario aplicar técnicas de preprocesamiento o usar métricas robustas.

```
sns.kdeplot(
    df.exam_score,
    fill =True,
    color = 'blue',
)
sns.rugplot(
    df.exam_score,
    color ='blue'
)
```

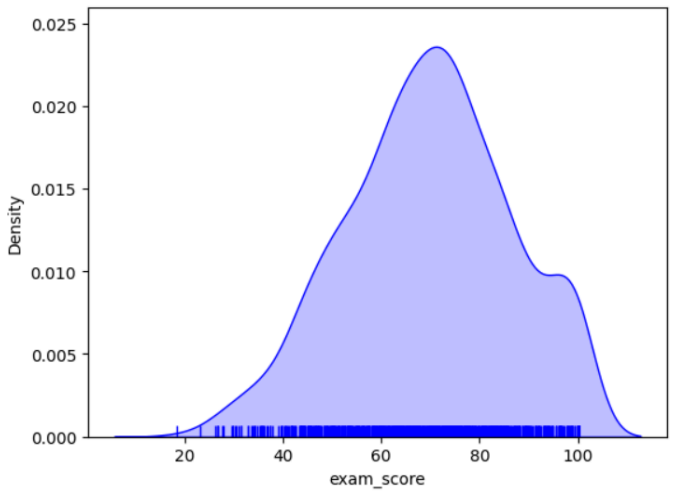

Fig 5. Distribución de la variable respuesta.

Se observa un sesgo positivo en la distribución de la variable, con los datos fuertemente concentrados en torno a la media y una cola extendida hacia valores mayores. Dado que esta es la variable objetivo que buscamos predecir, resulta conveniente que su distribución sea lo más cercana posible a una distribución normal, ya que esto permite un mejor control del modelo y una mejora en su rendimiento.

Para abordar esta asimetría, aplicamos técnicas de normalización, específicamente dos transformaciones comunes para corregir el sesgo:

- Transformación logarítmica
- Transformación de raíz cuadrada

Estas transformaciones se utilizaron con el fin de comparar su capacidad para reducir el sesgo y aproximar la distribución a la normalidad, lo cual evaluaremos visual y estadísticamente.

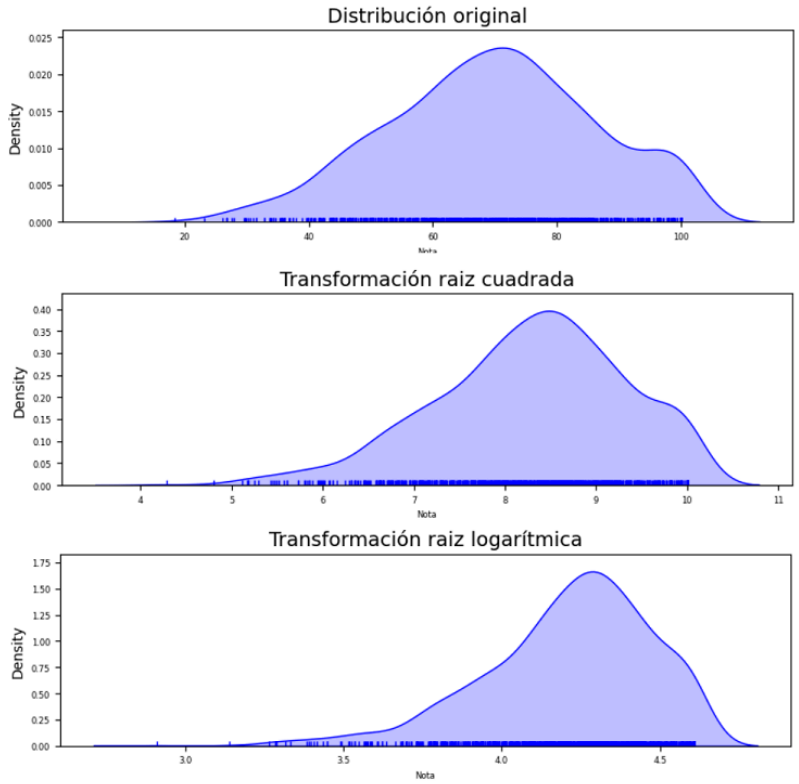

Fig 6. Comparación de la dist. de la variable respuesta con diferentes transf.

Para hacer una prueba a que distribución se aproxima más nuestra variable respuesta usamos fitter:

```
dist = ['chi2','cauchy', 'exponpow','expon','gamma','norm','powerlaw','beta','logistic']
fitter = Fitter(df.exam_score, distributions=dist)
fitter.fit()
fitter.summary(Nbest=10,plot = False)
```

	sumsquare_error	aic	bic	kl_div	ks_statistic	ks_pvalue
norm	0.005190	985.702409	995.517920	inf	0.035864	1.490117e-01
gamma	0.005201	993.216125	1007.939391	inf	0.039076	9.189001e-02
chi2	0.005255	996.480514	1011.203780	inf	0.041500	6.206239e-02
logistic	0.005271	981.866178	991.681688	inf	0.043740	4.229109e-02
beta	0.005544	972.664744	992.295765	inf	0.054228	5.370036e-03
cauchy	0.006324	1008.839400	1018.654911	inf	0.106730	2.252889e-10
powerlaw	0.007239	932.140185	946.863451	inf	0.142039	4.573714e-18
exponpow	0.015709	967.897851	982.621116	inf	0.299856	3.104356e-80
expon	0.018121	950.524097	960.339607	inf	0.331476	2.035064e-98

```
print(fitter.get_best())
```

```
{'norm': {'loc': 69.6015, 'scale': 16.880117527730665}}
```

Los resultados del análisis realizado con la librería fitter indican que la distribución normal es la que mejor se ajusta a nuestra variable exam_score. Esta conclusión se sustenta en varios criterios estadísticos:

- Suma de los errores cuadráticos: la distribución normal presenta el menor valor, lo que indica un mejor ajuste general a los datos observados.
- Criterios de información de Akaike (AIC) y Bayesiano (BIC): ambos son más bajos para la distribución normal, lo que sugiere que, además de ajustar bien, lo hace con menor complejidad, siendo penalizada en menor medida por posibles sobreajustes.
- Estadístico de Kolmogorov-Smirnov (KS): este test compara la función de distribución acumulada empírica con la teórica, y en este caso, la distribución normal arrojó el valor más bajo, lo que indica una mayor similitud entre ambas curvas.

Estos indicadores nos permiten concluir que la distribución normal es la mejor candidata para modelar exam_score, y respaldan el uso de transformaciones destinadas a aproximar la variable a dicha distribución.

Distribución de variables cualitativas

```
fig, axes = plt.subplots(nrows=2,ncols=3, figsize =(10,5))
axes = axes.flat #para poder usar el for con los plots
col_cualitativas = dfCat.columns

for i, colum in enumerate(col_cualitativas):
  dfCat[colum].value_counts().plot.barh(ax = axes[i])
  axes[i].set_title(colum, fontsize = 8, fontweight='bold')
  axes[i].tick_params(labelsize = 7)
  axes[i].set_xlabel("conteo")
fig.tight_layout
plt.subplots_adjust(hspace=0.4, wspace=0.4)
```

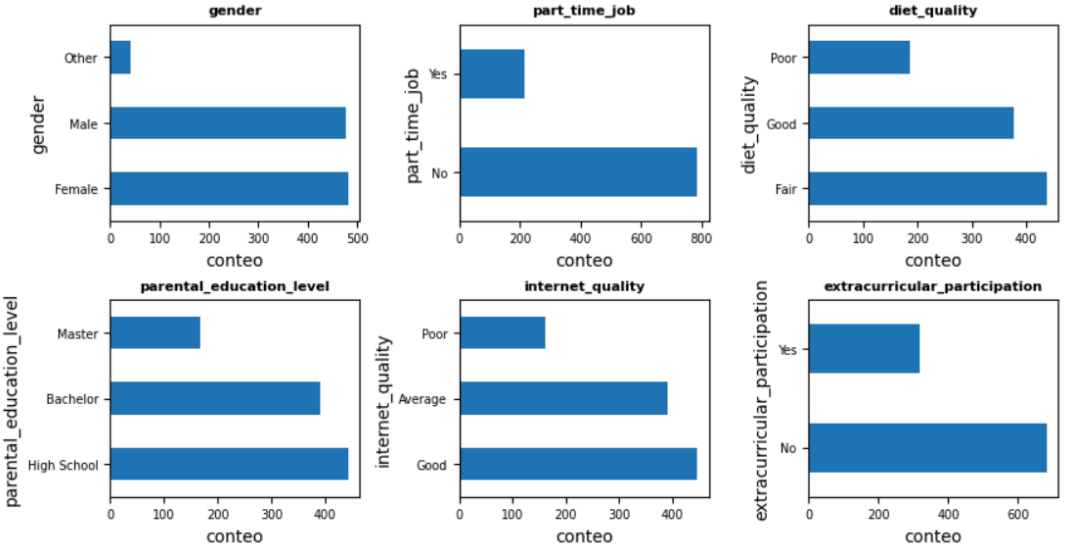

Fig 7. Distribución de las variables cat. (gráficos de barra).

Relación entre las notas y las variables cualitativas

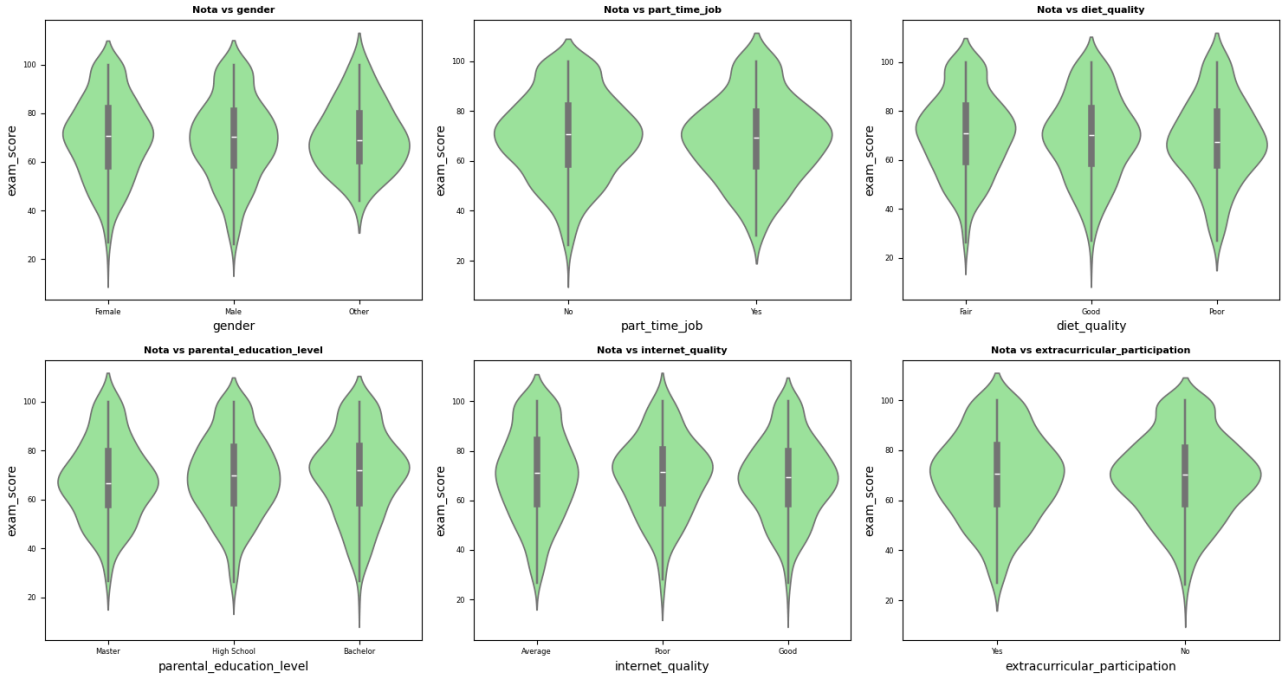

Fig 8. gráficos de caja 'exam_score' vs variables cualitativas junto con sus distribuciones.

Relación de variables categóricas con los puntajes de examen

Nos interesa analizar si existen diferencias significativas en los puntajes de examen (exam_score) entre los distintos grupos dentro de nuestras variables categóricas. En otras palabras, buscamos responder preguntas como:

- ¿Los estudiantes que trabajan media jornada obtienen, en promedio, una nota menor que aquellos que no trabajan?

Para ello, se aplican diferentes pruebas estadísticas de comparación de medias según la cantidad de categorías que presente cada variable:

- Para variables categóricas con dos niveles (por ejemplo, trabaja vs. no trabaja), se utiliza el test t de Student para muestras independientes. Este test permite determinar si las diferencias entre las medias de los grupos son estadísticamente significativas.
- Para variables con tres o más categorías (por ejemplo, nivel educativo de los padres), se recurre al Análisis de Varianza (ANOVA). Esta técnica evalúa si al menos una de las medias difiere significativamente del resto.

Estas pruebas nos permitirán establecer si pertenecer a una determinada categoría tiene un efecto relevante sobre el desempeño académico, medido a través de exam_score.

```
var_objetivo = 'exam_score'
var_categoricas = ['diet_quality', 'parental_education_level', 'internet_quality',
            'part_time_job', 'extracurricular_participation', 'gender']

resultados = []
```

```
for var in var_categoricas:
    grupos = df.groupby(var)[var_objetivo]
    categorias = grupos.groups.keys()
    print(f"\nAnalizando variable: {var} - categorías: {list(categorias)}")

    if len(categorias) == 2:
        grupo1 = grupos.get_group(list(categorias)[0])
        grupo2 = grupos.get_group(list(categorias)[1])
        print(f"Tamaño grupos: {len(grupo1)} y {len(grupo2)}")
        stat, p = ttest_ind(grupo1, grupo2)
        test_name = 't-test'
    elif len(categorias) > 2:
        valores = [grupo for _, grupo in grupos]
        print(f"Cantidad de grupos: {len(valores)}")
        stat, p = f_oneway(*valores)
        test_name = 'ANOVA'
    else:
        print("No se puede comparar, solo hay una categoría")
        continue

    print(f"p-valor: {p:.4f}")
    resultados.append({
        'variable': var,
        'test': test_name,
        'p_valor': p,
        'significativo': p < 0.05
    })
```

```
Analizando variable: diet_quality - categorías: ['Fair', 'Good', 'Poor']
Cantidad de grupos: 3
p-valor: 0.2824

Analizando variable: parental_education_level - categorías: ['Bachelor', 'High School', 'Master']
Cantidad de grupos: 3
p-valor: 0.3412

Analizando variable: internet_quality - categorías: ['Average', 'Good', 'Poor']
Cantidad de grupos: 3
p-valor: 0.2320

Analizando variable: part_time_job - categorías: ['No', 'Yes']
Tamaño grupos: 785 y 215
p-valor: 0.4006

Analizando variable: extracurricular_participation - categorías: ['No', 'Yes']
Tamaño grupos: 682 y 318
p-valor: 0.9778

Analizando variable: gender - categorías: ['Female', 'Male', 'Other']
Cantidad de grupos: 3
p-valor: 0.8674
```

Estos p-valores altos (todos mayores a 0.05) indican que no hay evidencia estadísticamente significativa de que las variables categóricas están asociadas con el puntaje del examen (exam_score).

La nota (exam_score) se comporta de forma similar, sin importar el género, si tienen trabajo de medio tiempo, la calidad del internet, etc. Puede que estas variables no influyen directamente en el rendimiento académico medido por ese puntaje. O bien, puede que influyen indirectamente o en combinación con otras que no estamos incluyendo (por ejemplo, salud mental, horas de estudio, motivación, etc.)

Análisis de Variables Numéricas

Antes que nada se realizaron los gráficos de histograma para tener una idea de la distribución de las variables numéricas o cuantitativas.

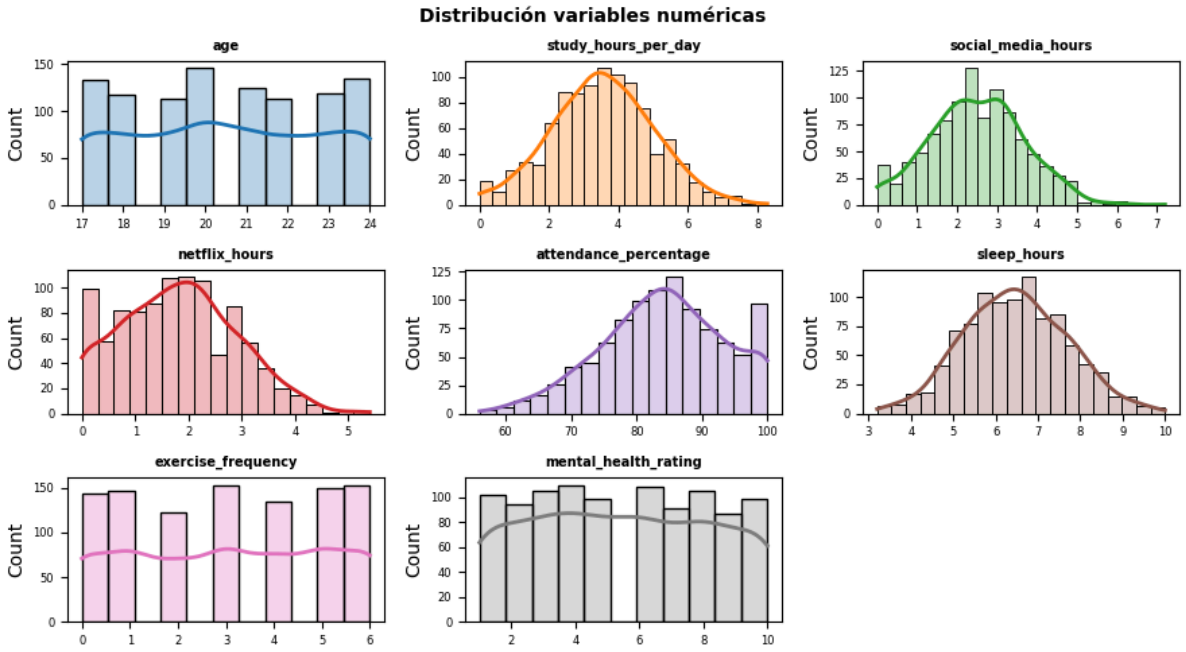

Fig 9. Distribución de las variables cuantitativas.

Como el objetivo del estudio es predecir la nota de examen, el análisis de cada variable se hace también en relación a la variable respuesta 'exam_score'. Analizando los datos de esta forma, se pueden extraer ideas sobre qué variables están más relacionadas con la nota de examen y de qué forma.

```python
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(9, 5))
axes = axes.flat
columnas_numeric = df.select_dtypes(include=['float64', 'int']).columns
columnas_numeric = columnas_numeric.drop('exam_score')

for i, colum in enumerate(columnas_numeric):
    sns.regplot(
        x           = df[colum],
        y           = df['exam_score'],
        color       = "gray",
        marker      = '.',
        scatter_kws = {"alpha":0.4},
        line_kws    = {"color":"r","alpha":0.7},
        ax          = axes[i]
    )
    axes[i].set_title(f"exam_score vs {colum}", fontsize = 7, fontweight = "bold")
    axes[i].tick_params(labelsize = 6)
    axes[i].set_xlabel("")
    axes[i].set_ylabel("")

# Se eliminan los axes vacíos
for i in [8]:
    fig.delaxes(axes[i])

fig.tight_layout()
plt.subplots_adjust(top=0.9)
fig.suptitle('Correlación con exam_score', fontsize = 10, fontweight = "bold");
```

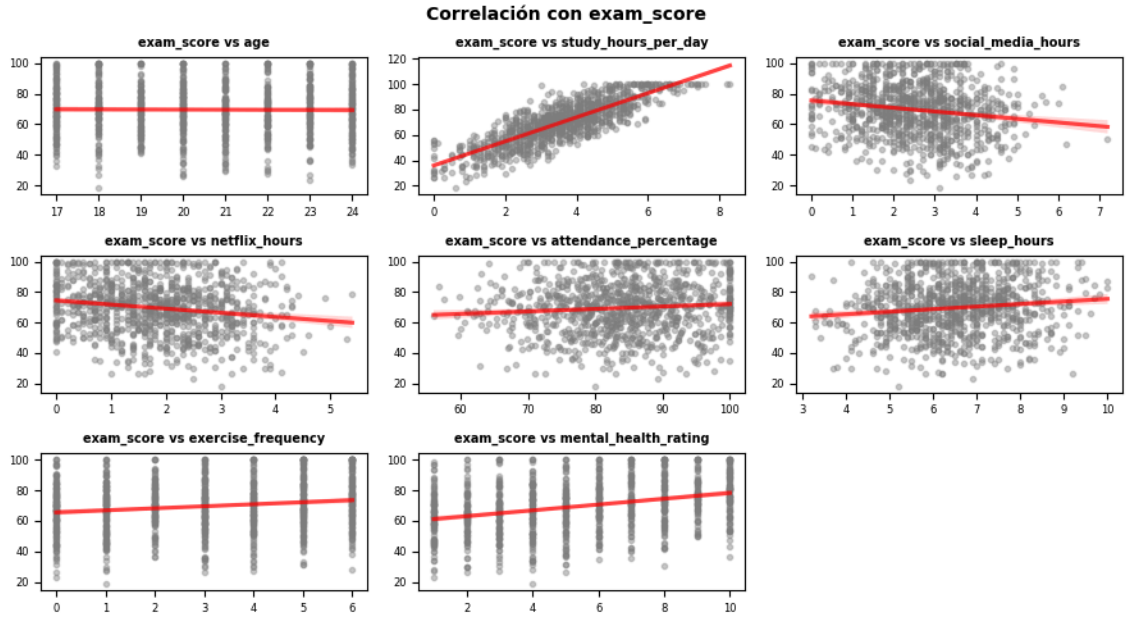

Fig 10. Regplots de exam_score vs variables numéricas.

Como algunas de las variables predictoras pueden ser colineales o estar correlacionadas y esto afecta la precisión, ya que me aportan información repetida de los modelos de regresión, pueden desencadenar los siguientes problemas:

- **Los coeficientes del modelo se vuelven inestables.** Pequeños cambios en los datos pueden provocar grandes cambios en los coeficientes
- **Dificulta la interpretación de los coeficientes.** No se sabe cuál variable está "explicando" el efecto.
- **Puede ocultar relaciones reales.** Algunas variables pueden parecer poco importantes sólo porque están solapadas con otras.
- **Puede provocar que el modelo no sea generalizable.** Es decir, puede funcionar bien en el entrenamiento, pero mal en nuevos datos.

Correlación entre las variables numéricas

Intentaremos ver qué coeficiente de correlación de Pearson tienen las variables. Para que sea más legible, se hace con el formato Tidy

```
def tidy_corr_matrix(corr_mat):

    corr_mat = corr_mat.stack().reset_index()
    corr_mat.columns = ['variable_1','variable_2','r']
    corr_mat = corr_mat.loc[corr_mat['variable_1'] != corr_mat['variable_2'], :]
    corr_mat['abs_r'] = np.abs(corr_mat['r'])
    corr_mat = corr_mat.sort_values('abs_r', ascending=False)

    return(corr_mat)

corr_matrix = df.select_dtypes(include=['float64', 'int']).corr(method='pearson')
tidy_corr_matrix(corr_matrix).head(10)
```

	variable_1	variable_2	r	abs_r
73	exam_score	study_hours_per_day	0.825419	0.825419
17	study_hours_per_day	exam_score	0.825419	0.825419
79	exam_score	mental_health_rating	0.321523	0.321523
71	mental_health_rating	exam_score	0.321523	0.321523
35	netflix_hours	exam_score	-0.171779	0.171779
75	exam_score	netflix_hours	-0.171779	0.171779
74	exam_score	social_media_hours	-0.166733	0.166733
26	social_media_hours	exam_score	-0.166733	0.166733
62	exercise_frequency	exam_score	0.160107	0.160107
78	exam_score	exercise_frequency	0.160107	0.160107

Fig 11. Correlación entre variables numéricas en formato Tidy.

De esta forma, obtuvimos un "Ranking" de las variables que están más fuertemente correlacionadas.

Lo cual nos da un pantallazo para visualizar que ningún r está por encima de 0,8 a excepción de study_hours_per_day con exam_score.

Matriz de correlación

```
plt.figure(figsize=(10, 6))
corr = df.select_dtypes(include=['float64', 'int']).corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlación entre Variables Numéricas')
plt.show()
```

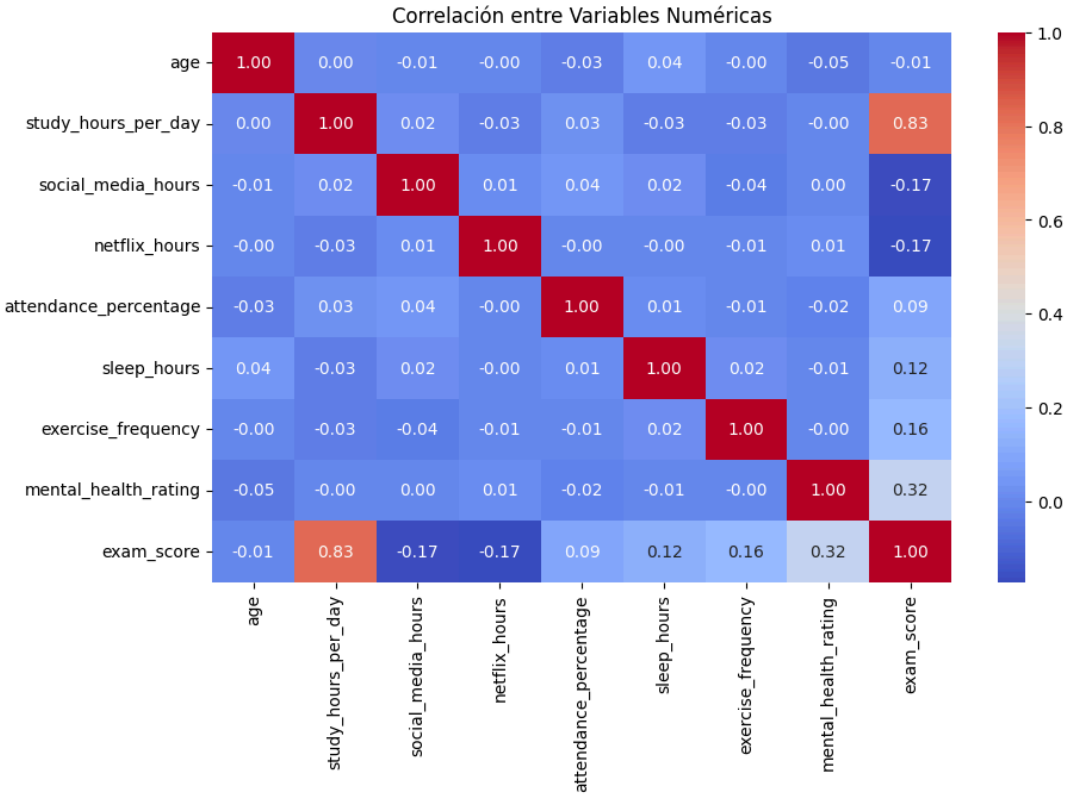

Fig 12. Matriz de correlación entre variables numéricas.

Un coeficiente de 0.83 entre Study_hours_per_day y exam_score indica una relación lineal fuerte y positiva, conforme aumentan las horas de estudio diarias , tienden a subir las notas de examen.

MODELADO

Preparación del Dataset para realizar el modelo supervisado (Random Forest)

Variables categóricas que debemos codificar:

- gender (Male/Female)
- part_time_job (Yes/No)
- diet_quality (Poor/Fair/Good)
- parental_education_level (High School, Master...)
- internet_quality (Poor/Average/Good)
- extracurricular_participation (Yes/No)

Usamos OneHotEncoder ya que las características de las variables no siguen un orden y queremos evitar introducir errores.

```
df = df.drop(columns=["student_id"])

# separamos entre predictores y variable objetivo
X = df.drop('exam_score',axis = 1)
y = df['exam_score']

columnas_cat = [
    'gender', 'part_time_job', 'diet_quality',
    'parental_education_level', 'internet_quality',
    'extracurricular_participation'
]

# Aplicamos OneHotEncoding para 'Vectorizar' nuestras columnas categoricas

procesamiento = ColumnTransformer(
    transformers = [('categoria', OneHotEncoder(drop="first"), columnas_cat)],
    remainder='passthrough'
)
```

Para este proyecto utilizamos **Pipeline** el cual es un flujo estructurado y automatizado que encadena todos los pasos necesarios para entrenar y evaluar un modelo, desde el preprocesamiento de los datos hasta la predicción final. Su objetivo es garantizar que cada paso se realice siempre de la misma forma, con reproducibilidad, limpieza y facilidad para probar distintos modelos o configuraciones.

```python
# Creamos un pipeline porque es mas facil llamarlo en vez de hacer la vectorizacion para cada columna cat

pipeline = Pipeline(steps=[
    ("Tratamiento", procesamiento),
    ("regressor", RandomForestRegressor(random_state=42))
])

# Separamos en train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Entrenamos el modelo
pipeline.fit(X_train, y_train)

# Predecimos
y_pred = pipeline.predict(X_test)
```

Evaluación del modelo

```python
from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"MSE: {mse:.2f}")
print(f"R^2: {r2:.2f}")
```

```
MSE: 38.51
R^2: 0.85
```

Un R^2 de 0.85 Significa que el 85% de la variabilidad de las notas puede explicarse por el modelo usando los hábitos como variables. Más en el ámbito de problemas educativos o sociales, donde hay alta variabilidad y ruido, un $R^2 > 0.8$ es considerado muy bueno.
A su vez el MSE el cual se refiere al promedio de los errores al cuadrado, un valor de 38.51 es bastante aceptable ya que nos está diciendo que en promedio tenemos un error de 6.2 puntos por nota.
Luego de esto realizamos comparaciones con las regularizaciones de Lasso(L1) y Ridge(L2) para compararlas con RF y ver cual es el mejor estimador.

Esto puede verse al detalle en Google Collaboratory.

Análisis con FairLearn

Para el siguiente análisis buscamos ver si el modelo favorece o perjudica más a ciertos grupos. Es decir, vamos a analizar la equidad con Fairlearn sobre nuestro mejor modelo, usando las variables más sensibles. En este contexto, las más comunes suelen ser:

- gender
- parental_education_level
- internet_quality
- mental_health_rating

Estas variables pueden reflejar condiciones sociales o estructurales que podrían estar asociadas a desigualdades.

```python
###Variables sensibles a evaluar
sensitive_features = {
    "Gender": X_test["gender"],
    "Parental Education": X_test["parental_education_level"],
    "Internet Quality": X_test["internet_quality"],
    "Mental Health Rating": X_test["mental_health_rating"]
}

# Evaluamos con las métricas clásicas
metrics = {
    "MSE": mean_squared_error,
    "R²": r2_score
}

# Diccionario para guardar resultados por grupo
resultados_equidad = {}

for nombre, feature in sensitive_features.items():
    mf = MetricFrame(
        metrics=metrics,
        y_true=y_test,
        y_pred=mejor_modelo.predict(X_test),
        sensitive_features=feature
    )
    resultados_equidad[nombre] = mf
    print(f"\n Métricas por grupo para: {nombre}")
    print(mf.by_group)

    # Gráfico
    mf.by_group.plot(kind="bar", title=f"Métricas por grupo: {nombre}", figsize=(8, 4))
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

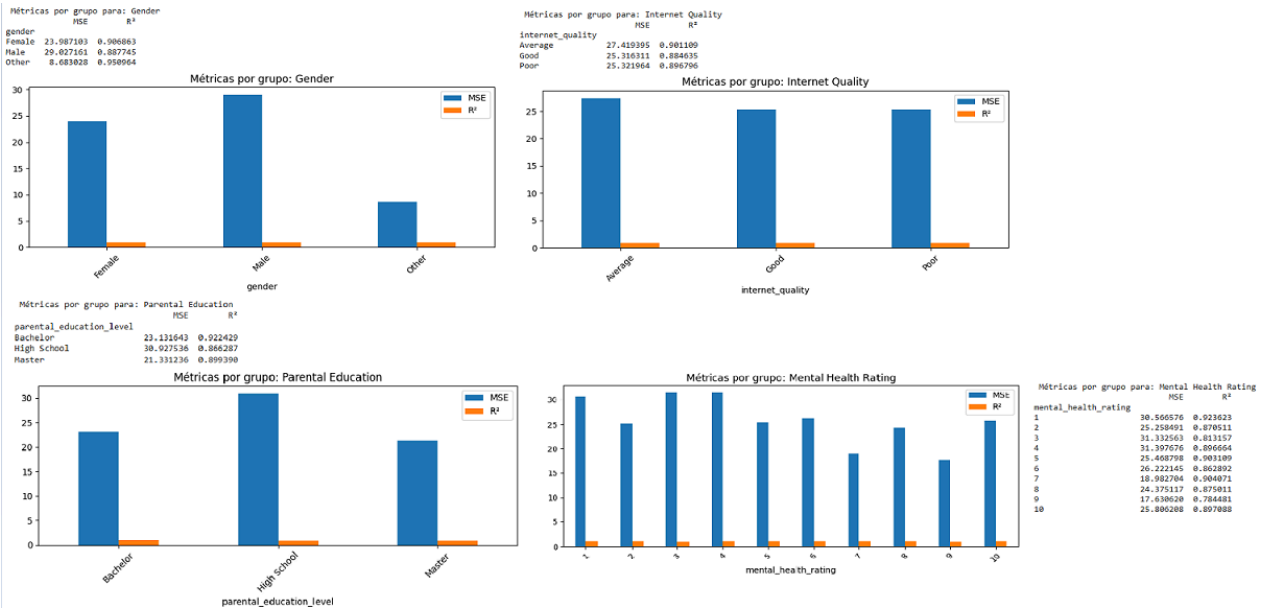

Fig 13. Métricas por variables sensibles. Ver en Colab.

Conclusiones sobre Fairlearn

1. **Género (Gender)**:
    - MSE (Error cuadrático medio): El modelo es más preciso para "Other", con un error más bajo. Tiene un peor rendimiento para "Male".
    - R^2 (Coef. de determinación): Aunque todos los valores son buenos (>0.88), hay una pequeña diferencia entre "Other" (0.95) y "Male" (0.88).

   Conclusión: No hay disparidades graves, pero podríamos monitorear el desempeño en "Male", que tiene el peor error. El grupo "Other" es pequeño probablemente, lo cual también puede sesgar esta métrica.

2. **Nivel educativo de los padres (Parental Education):** El modelo rinde peor en estudiantes con padres que tienen solo secundaria, tanto en MSE como en R^2.
   Las mejores métricas son para quienes tienen padres con estudios superiores.
   Conclusión: Hay disparidad notable. El modelo es menos preciso para estudiantes con menor nivel educativo en el hogar, lo cual puede reforzar desigualdades.
3. **Calidad de Internet (Internet Quality):** Muy consistente entre grupos. Las diferencias en MSE y R^2 son mínimas.
   Conclusión: No hay disparidad significativa por calidad de internet.
4. **Salud mental (Mental Health Rating):** Claramente, el modelo predice peor para estudiantes con puntajes medios de salud mental (ej.: 3 y 4). Predice mejor para puntajes altos (7, 9).
   Conclusión: Posible sesgo relacionado al estado emocional o bienestar. Sería interesante ver si estas categorías están correlacionadas con otras variables (como sueño, dieta, etc.).

Probando Modelos con Pycaret

¿Qué hace PyCaret?

PyCaret automatiza muchas tareas comunes del flujo de trabajo de machine learning, como:

- Preprocesamiento de datos (limpieza, codificación, escalado, imputación, etc.)
- Entrenamiento de múltiples modelos con una sola línea de código.
- Comparación automática de modelos.
- Selección del mejor modelo.
- Optimización de hiper parámetros.
- Interpretación del modelo.

- Guardado y despliegue del modelo.

También utilizamos las librerías de Mlflow, Flask y Ngrok la cual nos permite utilizar un proxy reverso ya que estamos trabajando con el entorno de Google Collaboratory.

Ngrok:

Ngrok es una herramienta que nos permite crear un túnel seguro desde internet a nuestra computadora o servidor local. Es como abrir una puerta temporal en internet para acceder a algo que estás ejecutando localmente.

¿Para qué sirve?

Por ejemplo, si tenemos una aplicación web o un servidor en nuestra máquina o en Google Colab, con Ngrok podemos obtener un link público (como https://xyz.ngrok.io) para acceder a esa app desde cualquier navegador, incluso desde el celular o mostrarla a otra persona.

Mlflow:

Es una app o servicio que permite hacer todo el tracking de los modelos de Machine Learning desde el inicio hasta su despliegue. Es decir, nos ayuda a gestionar el ciclo de vida completo de un modelo de Machine Learning.

FLask:

Vamos a necesitar flask para el despliegue ya que haremos una API para poder hacer consultas con nuestro modelo, para esto necesitamos un web service o pequeño servidor web donde podamos alojar y consultar nuestra API esto nos permitirá hacer flask.

¿Dónde guardamos los datos de Mlflow?

Como estamos usando Google Colab que está en el ecosistema de Google los guardamos en el Google Drive:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Definimos dónde MLflow guardará todo:
MLFLOW_ROOT = "/content/drive/MyDrive/mlflow_runs"
```

Configuramos Mlflow.

```
import mlflow
mlflow.set_tracking_uri(f"file://{MLFLOW_ROOT}")
```

Arrancamos Mlflow.

```
import subprocess, time
mlflow_server = subprocess.Popen([
    "mlflow", "ui",
    "--backend-store-uri", f"file://{MLFLOW_ROOT}",
    "--port", "5000"
])
time.sleep(5)

# Usamos NGROK para proxy reverso y poder acceder.
from pyngrok import ngrok, conf
import getpass

print("Ingresar auth token de ngrok:  https://dashboard.ngrok.com/auth")
conf.get_default().auth_token = getpass.getpass()

port = 5000
public_url = ngrok.connect(port).public_url
print(f' * ngrok tunnel "{public_url}" -> "http://127.0.0.1:{port}"')
```

```
Ingresar auth token de ngrok:  https://dashboard.ngrok.com/auth
··········
 * ngrok tunnel "https://314a-34-21-56-251.ngrok-free.app" -> "http://127.0.0.1:5000"
```

Antes de configurar y usar Pycaret debemos preparar nuevamente nuestro dataset.

```
    # Definimos columnas categóricas
    categorical_cols = [
        "gender",
        "part_time_job",
        "diet_quality",
        "parental_education_level",
        "internet_quality",
        "extracurricular_participation"
    ]


    # Aplicar One-Hot Encoding (NO eliminamos ninguna categoría)
    df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_first=False)

    # Convertir posibles valores booleanos a enteros (True/False → 1/0)
    df_encoded = df_encoded.astype(int)

    # Mostrar las primeras filas para verificar
    print(df_encoded.head())
    print(df_encoded.columns.tolist())
```

Cargamos y configuramos PyCaret con Mlflow para hacer el seguimiento.

```
from pycaret.regression import setup, compare_models, finalize_model, save_model

#Creamos el experimento en MLflow
mlflow.set_experiment("Predicciones con el dataset Desempeño académico de los estudiantes en base a sus hábitos2")

exp = setup(
    data=df_encoded,
    target='exam_score',
    session_id=42,
    log_experiment=True,
    experiment_name='Predicciones con el dataset Desempeño académico de los estudiantes en base a sus hábitos2',
    log_plots=True          # registra gráficos (conf matrix, etc.)
)
```

Entrenamos y comparamos los modelos.

```
best = compare_models()

# Finaliza el modelo para habilitar predicciones en producción
model = finalize_model(best)

# Guarda localmente (un .pkl) y también registra en MLflow
save_model(model, 'modelo_Estudiantes')
```

Resultados arrojados por Pycaret al analizar distintos modelos para nuestros datos

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
br	Bayesian Ridge	4.3295	29.9777	5.4465	0.8896	0.0838	0.0670	0.0280
lr	Linear Regression	4.3323	30.0160	5.4498	0.8895	0.0838	0.0670	1.1370
ridge	Ridge Regression	4.3316	30.0056	5.4489	0.8895	0.0838	0.0670	0.0410
lar	Least Angle Regression	4.3323	30.0160	5.4498	0.8895	0.0838	0.0670	0.0930
huber	Huber Regressor	4.3195	30.1916	5.4650	0.8888	0.0838	0.0666	0.0640
lasso	Lasso Regression	4.4979	32.4528	5.6706	0.8818	0.0895	0.0708	0.0530
llar	Lasso Least Angle Regression	4.4979	32.4528	5.6706	0.8818	0.0895	0.0708	0.0600
gbr	Gradient Boosting Regressor	4.6449	34.5310	5.8471	0.8717	0.0938	0.0735	0.1440
lightgbm	Light Gradient Boosting Machine	4.7879	36.1898	5.9897	0.8657	0.0954	0.0754	0.1890
en	Elastic Net	5.2019	42.2582	6.4796	0.8478	0.1060	0.0843	0.0770
rf	Random Forest Regressor	5.1916	42.2605	6.4678	0.8452	0.1044	0.0827	0.4160

Registramos el mejor modelo en Mlflow.

```
# Registrar en MLflow
mlflow.sklearn.log_model(model,artifact_path="modelo_Estudiantes") # Mejor Modelo para prod
```

Debemos indicar también donde ha quedado guardado nuestro modelo para así poder usarlo en la API.

```
run_id = mlflow.last_active_run().info.run_id
model_uri = f"runs:/{run_id}/modelo_Estudiantes"
print("Model URI:", model_uri)
```

```
Model URI: runs:/35364e4131094b159690b5fabd7a3134/modelo_Estudiantes
```

Creamos la Api.

```
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route("/predict", methods=["POST"])
def predict():
    try:
        data = request.get_json()
        df = pd.DataFrame(data['data'], columns=data['columns'])

        pred = model.predict(df)
        return jsonify({"prediction": pred.tolist()})
    except Exception as e:
        return jsonify({"error": str(e)})
```

```
from pyngrok import ngrok
import threading

# Crear un túnel a la API Flask
public_url = ngrok.connect(5002)
print(f"🚀 Tu API Flask está disponible en: {public_url}/predict")

# Ejecutar Flask en segundo plano
def run_flask():
    app.run(port=5002)

thread = threading.Thread(target=run_flask)
thread.start()
```

```
🚀 Tu API Flask está disponible en: NgrokTunnel: "https://aa83-34-21-56-251.ngrok-free.app" -> "http://localhost:5002"/predict
```

Prueba de la Api

Los datos deben enviarse en el mismo orden con el que fueron entrenados (obviamente sin exam_score):

['age', 'study_hours_per_day', 'social_media_hours', 'netflix_hours', 'attendance_percentage', 'sleep_hours', 'exercise_frequency', 'mental_health_rating', 'exam_score', 'gender_Female', 'gender_Male', 'gender_Other', 'part_time_job_No', 'part_time_job_Yes', 'diet_quality_Fair', 'diet_quality_Good', 'diet_quality_Poor', 'parental_education_level_Bachelor', 'parental_education_level_High School', 'parental_education_level_Master', 'internet_quality_Average', 'internet_quality_Good', 'internet_quality_Poor', 'extracurricular_participation_No', 'extracurricular_participation_Yes']

```
lista = df_encoded.head(2).values.tolist()
print(lista)
```

```
[[23, 0, 1, 1, 85, 8, 6, 8, 56, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1], [20, 6, 2, 2, 97, 4, 6, 8, 100, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0]]
```

```python
import requests

sample = {
  "columns": [
    'age', 'study_hours_per_day', 'social_media_hours', 'netflix_hours', 'attendance_percentage', 'sleep_hours',
    'exercise_frequency', 'mental_health_rating', 'gender_Female', 'gender_Male', 'gender_Other', 'part_time_job_No',
    'part_time_job_Yes', 'diet_quality_Fair', 'diet_quality_Good', 'diet_quality_Poor', 'parental_education_level_Bache
    'parental_education_level_Master', 'internet_quality_Average', 'internet_quality_Good', 'internet_quality_Poor', 'e
  ],
  "data": [
    [23, 0, 1, 1, 85, 8, 6, 8, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1], # Nota les borre a mano el exam score
    [20, 6, 2, 2, 97, 4, 6, 8, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0]
  ]
}
response = requests.post("https://aa83-34-21-56-251.ngrok-free.app/predict", json=sample)
print("Status code:", response.status_code)
print("Response text:", response.text)  # Esto te muestra qué devolvió el servidor (aunque no sea JSON)
try:
    data = response.json()
    print("JSON response:", data)
except Exception as e:
    print("Error decodificando JSON:", e)
```

Entonces la API devuelve las siguientes predicciones para los dos primeros alumnos:

```
INFO:werkzeug:127.0.0.1 - - [29/May/2025 20:58:07] "POST /predict HTTP/1.1" 200 -
Status code: 200
Response text: {"prediction":[56.60895414159046,101.74224075523848]}

JSON response: {'prediction': [56.60895414159046, 101.74224075523848]}
```

Si nos fijamos el exam_score de los primeros 2 estudiantes:

```python
print(df_encoded['exam_score'].head(2))
```

```
0     56
1    100
Name: exam_score, dtype: int64
```

La predicción realizada por el mejor modelo seleccionado por PyCaret, en este caso Bayesian Ridge, estima que el primer alumno obtendría un puntaje de 56,6, mientras que el segundo alumno alcanzaría un 101,7. Si bien este último valor se encuentra fuera del rango típico de puntajes, es importante destacar que este comportamiento es esperado en modelos de regresión, especialmente en aquellos que se basan en distribuciones de probabilidad, como es el caso de la regresión bayesiana.

Esto no implica necesariamente un error del modelo, sino que refleja la naturaleza probabilística de sus predicciones. Sin embargo, en contextos reales, puede ser necesario ajustar o acotar los valores de salida para que se mantengan dentro de los límites esperados (por ejemplo, entre 0 y 100), dependiendo de cómo se utilizarán las predicciones en la práctica.

CONCLUSIONES

En el proceso de selección de modelos para predecir la nota de examen de alumnos en base a sus hábitos, se evaluaron diversos algoritmos, entre ellos Random Forest y Bayesian Ridge Regression. A pesar de que Random Forest es un modelo poderoso no lineal y robusto frente a outliers, en este caso Bayesian Ridge mostró un mejor desempeño general según las métricas de evaluación.

El dataset utilizado constaba de 1000 filas y 16 columnas, con variables relacionadas principalmente con hábitos de estudio, sueño, alimentación y asistencia. Estas características presentan relaciones predominantemente lineales o suavemente correlacionadas, lo cual favorece a modelos lineales con regularización como Bayesian Ridge.

Bayesian Ridge supera a Random Forest por las siguientes razones:

1. Tamaño del dataset:
    - Con solo 1000 muestras, Random Forest puede sobre ajustar fácilmente si no se regula bien.
    - Bayesian Ridge, al ser un modelo lineal con prior bayesiano, maneja mejor el sesgo-varianza y se adapta bien a conjuntos de datos pequeños o medianos.
2. Relaciones lineales entre variables:
    - Si los hábitos de los alumnos afectan de forma más o menos lineal a la nota (por ejemplo, más horas de estudio → mejor nota), entonces los modelos lineales como Bayesian Ridge pueden capturar mejor esa relación sin agregar complejidad innecesaria.
3. Regularización automática:
    - Bayesian Ridge incorpora una forma bayesiana de regularización (similares a Ridge/L2), lo cual reduce el sobreajuste y mejora la generalización.
    - Random Forest requiere una mayor cantidad de parámetros a ajustar y puede producir modelos más complejos sin una ganancia real de rendimiento.
4. Simplicidad y eficiencia:
    - Bayesian Ridge es computacionalmente más eficiente que Random Forest y facilita la interpretación de los coeficientes, lo cual es valioso en contextos educativos donde se busca comprender qué hábitos impactan más en el rendimiento.

Bayesian Ridge resultó ser el mejor estimador porque logró un equilibrio óptimo entre rendimiento, interpretabilidad y capacidad de generalización en un dataset con tamaño moderado y relaciones mayormente lineales entre las variables. Esto destaca la importancia de considerar tanto la naturaleza del problema como la complejidad del modelo al momento de seleccionar un algoritmo de machine learning.

BIBLIOGRAFÍA

Scikit-learn Regression Documentation. (n.d.). https://scikit-learn.org/stable/supervised_learning.html

Jake VanderPlas. Python Data Science Handbook. (n.d.).

Aurelien Geron. (3 ed.). Aprende Machine Learning con Scikit-Learn, Keras y TensorFlow

https://cienciadedatos.net/