

Universidad Tecnológica Nacional

Facultad Regional Córdoba

Ingeniería en Sistemas de Información



Ciencia de Datos

Ignite Your Mind (IYM)

“Tercera Entrega”

Grupo 3

Integrantes:

- 82217 Anzulovich Garzón, Valentina - valenanzu.utn@gmail.com
- 80359 Berrotarán, María Luz - luzberrotaran@gmail.com
- 78678 Salas, Bruno Matías - brunosalas1@hotmail.com
- 82508 Sonzini Astudillo, Enrique José - enriquesonzini@gmail.com
- 81841 Yarbouh, Yamili - yamiliyarbouh@gmail.com

Docentes:

- Pablo Alberto Sacco
- Franco Mana
- Marisa del Carmen Callejas

Curso: 5K2

Fecha de presentación: 30/10/2024

Índice

Índice.....	2
Introducción.....	3
Desarrollo.....	3
Modificación del Dataset.....	3
Las columnas de Lengua.....	3
Las columnas de matemáticas.....	3
Modelos.....	3
Modelo binario de clasificación.....	4
Árbol de decisión.....	4
Adaboost.....	4
RandomTreeClasifier.....	4
Modelo de regresión aplicado a un dataset binario.....	4
Conclusión.....	6

Introducción

En esta tercera entrega se desarrollaron y evaluaron diferentes modelos de predicción para identificar patrones asociados al rendimiento académico de estudiantes en las materias de Lengua y Matemática. El objetivo principal fue construir un sistema de alerta temprana que permita al colegio prever si un estudiante está en riesgo de no aprobar y, en consecuencia, intervenir con medidas de apoyo personalizadas.

Para alcanzar este objetivo, se llevó a cabo un análisis exhaustivo del dataset limpio, lo que incluyó el cálculo de correlaciones entre diversas variables y las calificaciones obtenidas en los exámenes. Como resultado de este análisis, se filtraron y eliminaron aquellas variables con baja correlación en cada caso para mejorar la precisión de los modelos. A continuación, se exploraron diferentes algoritmos de clasificación binaria, entre ellos el árbol de decisión, Adaboost, bosque aleatorio y regresión logística, cada uno adaptado y optimizado para predecir si un estudiante aprobará o no.

Por último evaluamos el desempeño de los modelos con métricas como precisión, exactitud, recall, F1-Score, curva ROC y AUC.

Desarrollo

Modificación del Dataset

Al generar los modelos de predicción, se descubrió que los mismos presentaban un alto nivel de imprecisión. Es por esto que se decidió someter al Dataset a dos análisis, uno para mpuntaje y otro para lpuntaje, centrándose en descubrir cuales son las columnas que más afectan, a cada una de las variables objetivo. La idea es ver los índices de correlación que tiene cada columna del dataset con la variable objetivo y eliminar aquellas cuyos valores absolutos son menores a 0.05.

Las columnas de Lengua

Las columnas del Dataframe cuyos valores de correlación con respecto a los puntajes de los exámenes de lengua fueron demasiado bajos como para ser considerados en el modelo fueron:

Columnas	Valores de correlación
ambito	-0.0034781819497568672
vive_padre	0.013233626297901223
vive_hermano	0.015574856442173914
tiene_tablet	0.01462273732959069
cuida_familiar	-0.021617657907866097
realiza_tareas_hogar	0.043255302429567856
dias_trabajo_fuera_casa	-0.03327619112791157
trabaja_remunerado	0.02538327905749644
fuera_escuela_deporte	-0.022856920717590767
faltas	0.025980018948553556
motivo_faltas_problemas_clima	0.03268154245415922
conflicto_trado_adultos	-0.019878256020575285
padres_extranjeros	-0.006795863795373289

Estas columnas fueron eliminadas para la creación del modelo de predicción de notas en los exámenes aprender de lengua.

Las columnas de matemáticas

Las columnas del Dataframe cuyos valores de correlación con respecto a los puntajes de matemáticas fueron demasiado bajos como para ser considerados en el modelo fueron:

Columnas	Valores de correlación
vive_padre	0.04738802794908249
vive_hermano	0.026832223214891683
tiene_tablet	0.04482508029445094
cuida_familiar	-0.040312238922684235
realiza_tareas_hogar	0.0006759291857699136
dias_trabajo_fuera_casa	0.028722421452655772
fuera_escuela_leer	0.043259533930424136
fuera_escuela_deporte	0.03234488867198128
fuera_escuela_clases_artes	0.048290062473476836
faltas	-0.03177040282172089
motivo_faltas_problemas_clima	-0.005785642798944287

conflicto_trado_adultos	0.0029140041796409536
conflicto_tratado_sanciones	0.035674166214867915
padres_extranjeros	0.0451607434081947

Estas columnas fueron eliminadas para la creación del modelo de predicción de notas en los exámenes aprender de matemáticas.

Modelos

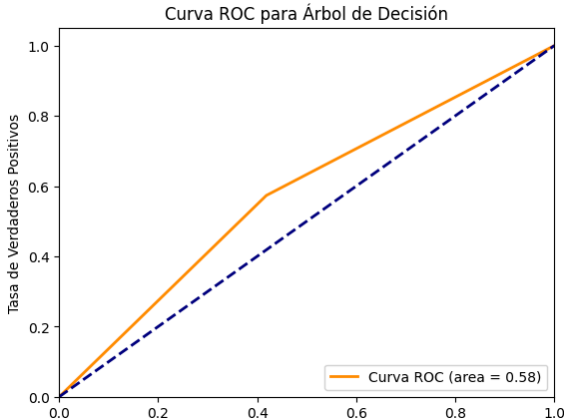
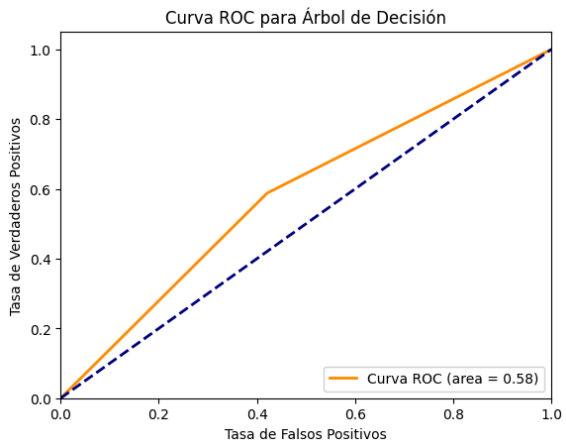
La idea original era crear un modelo de predicción en base a la regresión lineal que, a partir de las variables como sexo, edad o nivel educativo de los padres, lograrse predecir los resultados. Como la mayoría de los valores eran de naturaleza discreta, esto se volvió casi imposible. Por dicha razón, tuvimos que cambiar la metodología para armar un modelo binario, el cual predice a partir de las mismas variables si el alumno va a conseguir o no una nota mayor al promedio.

Como las precisiones de los modelos binarios varían dependiendo de su naturaleza (Árbol de decisión, Adaboost, Árbol de bosque aleatorio y regresión logística) tuvimos que crear un modelo por cada tipo de modelo para encontrar la relación entre los mismos. Vale la pena mencionar que se crearon dos modelos de cada tipo, uno para predecir los resultados de lengua y otro para predecir los resultados de matemáticas. Como fueron creados en instancias separadas, ambos tienen el mismo nombre, pero al ser dos modelos diferentes sus parámetros difieren.

Modelo binario de clasificación

Árbol de decisión

El primer modelo de clasificación elegido fue el de un árbol de decisión llamado clf. Elegimos este modelo porque es el más básico para trabajar con árboles binarios y creímos que sería una buena forma de iniciar con la experimentación. Además consideramos que el árbol de decisión podría servir de base para la generación de un modelo Adaboost.

Lengua	Matemáticas
Precisión: 0.6440379403794038 Exactitud: 0.6449746192893401 F1-score: 0.6444090431481544 Recall: 0.6447805738316489	Precisión: 0.6325918079096046 Exactitud: 0.6466879984717513 F1-score: 0.6595490105844455 Recall: 0.688905979619304
Curva ROC 	Curva ROC 
Matriz de confusión [8611 6196] [6290 8453]	Matriz de confusión [6105 4432] [4290 6112]

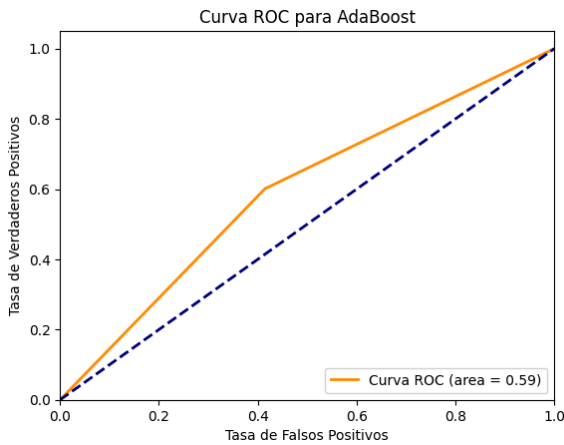
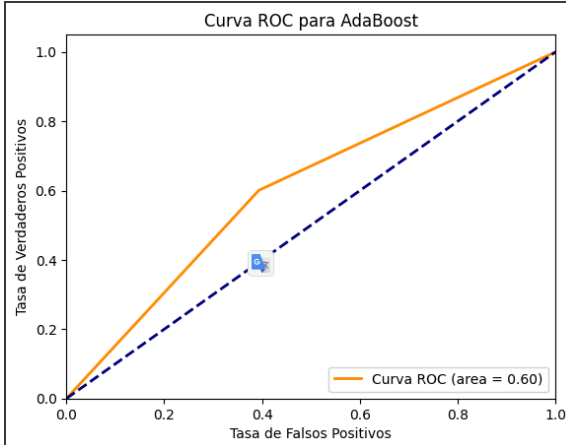
Validación cruzada	Validación cruzada
Precisión promedio: 0.6287	Precisión promedio: 0.5922
Desviación estándar: 0.0123	Desviación estándar: 0.0511
Hiperparámetros	Hiperparámetros
max_depth = 9	max_depth = 10

El modelo obtuvo una precisión del 64% para lengua y 63% para matemática lo que nos indica que existe una relación entre las columnas del dataset y el rendimiento académico del alumno en las materias de lengua y matemática y que el modelo la está detectando.

En términos de hiperparámetros para el modelo del árbol de decisión nos centramos en la profundidad máxima. Mientras que en lengua el mejor resultado se obtuvo con una profundidad de 9, en matemática el mejor resultado se logró con una profundidad de 10.

Adaboost

El segundo modelo elegido fue uno de tipo Adaboost, el cual usaría como base el árbol de decisión definido previamente.

Lengua	Matemáticas
Precisión: 0.6540514645496852 Exactitud: 0.653434856175973 F1-score: 0.6511326860841424 Recall: 0.6482398426371837	Precisión: 0.6531645569620254 Exactitud: 0.6592482926596304 F1-score: 0.6613186500213605 Recall: 0.6696789079023264
Curva ROC 	Curva ROC 
Matriz de confusión [8669 6138] [5883 8860]	Matriz de confusión [6391 4146] [4153 6249]

Validación cruzada Precisión promedio: 0.6266 Desviación estándar: 0.0144	Validación cruzada Precisión promedio: 0.5805 Desviación estándar: 0.0613
Hiperparámetros n_estimators = 50 learning_rate = 0.1	Hiperparámetros n_estimator= 50 learning_rate = 0.1

Con el modelo Adaboost obtuvimos una precisión menor que con el anterior, pero se sigue superando el 50% de precisión por lo que este modelo también detecta una correlación entre las columnas seleccionadas del dataset y el rendimiento académico del alumno.

En el caso de Adaboost analizamos 3 valores de estimadores 50, 100, 150 y 3 posibilidades de ritmos de aprendizaje 0.1, 0.5, 1.0. Tanto en lengua como en matemática la mejor opción fue trabajar con 50 estimadores y un ritmo de aprendizaje de 0.1.

Bosque aleatorio

En tercer lugar, elegimos crear otro modelo de tipo bosque aleatorio. Con este pensábamos alejarnos de la base árbol de decisión con la esperanza de encontrar resultados más satisfactorios.

Lengua	Matemáticas
Precisión: 0.6616394358329842 Exactitud: 0.657766497461929 F1-score: 0.6520557371408912 Recall: 0.6427457098283932	Precisión: 0.6624142263458007 Exactitud: 0.6637375232819142 F1-score: 0.660658344980481 Recall: 0.6589117477408191
Curva ROC 	Curva ROC
Matriz de confusión [9961 4846] [5267 9476]	Matriz de confusión [7044 3493] [3548 6854]

Validación cruzada Precisión promedio: 0.6450 Desviación estándar: 0.0072	Validación cruzada Precisión promedio: 0.6015 Desviación estándar: 0.0621
Hiperparámetros n_estimators = 100 max_depth = 10 min_samples_splits = 10	Hiperparámetros n_estimators = 500 max_depths = 20 min_samples_splits = 10

Este modelo funcionó muy bien en ambos casos, superando la precisión base del 65% en ambos modelos. La validación cruzada de lengua no llegó a superar el 65% por muy poco margen, y la de matemáticas con suerte alcanzó a superar el 60%.

Para Random Forest probamos variar el número de estimadores, las profundidades máximas y el mínimo de divisiones de las muestras. Para esto consideramos los siguientes valores:

n_estimators_list = [100, 300, 500]

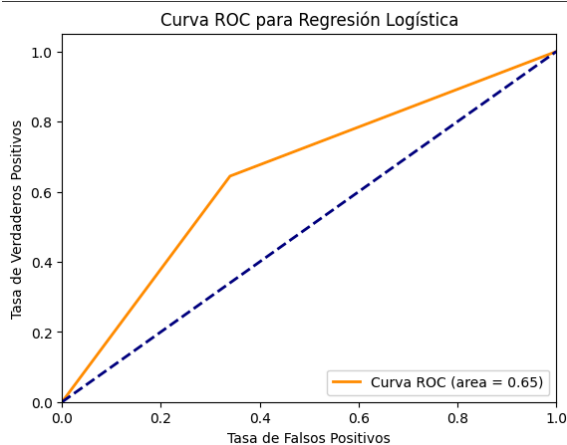
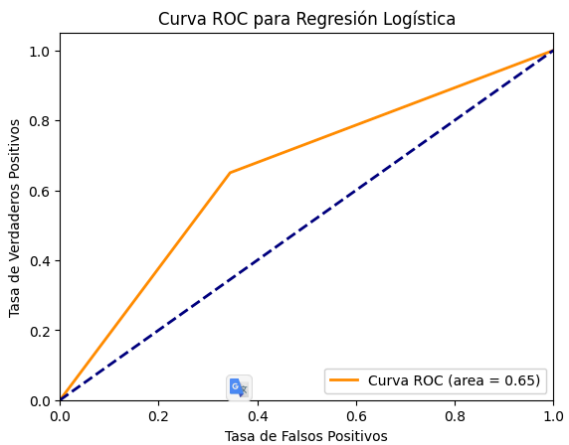
max_depths = [None, 10, 20]

min_samples_splits = [2, 5, 10]

En este caso obtuvimos diferentes resultados de cuáles hiperparámetros eran mejores para matemática o lengua.

Modelo de regresión logística

En cuarto lugar, intentamos aplicar un modelo de regresión logística binaria.

Lengua	Matemáticas
<p>Precisión: 0.6536981028320044</p> <p>Exactitud: 0.652419627749577</p> <p>F1-score: 0.6493462155610938</p> <p>Recall: 0.645051889032083</p>	<p>Precisión: 0.650538668718738</p> <p>Exactitud: 0.6527054778165147</p> <p>F1-score: 0.6503509952880084</p> <p>Recall: 0.6501634301095943</p>
<p>Curva ROC</p> 	<p>Curva ROC</p> 
<p>Matriz de confusión</p> <p>[9775 5032]</p> <p>[5243 9500]</p>	<p>Matriz de confusión</p> <p>[6903 3634]</p> <p>[3639 6763]</p>
<p>Validación cruzada</p> <p>Precisión promedio: 0.6431</p>	<p>Validación cruzada</p> <p>Precisión promedio: 0.6351</p>

Desviación estándar: 0.0062	Desviación estándar: 0.0281
Hiperparámetros C_value = 0.001	Hiperparámetros C_value = 0.01

Este modelo alcanzó una precisión base de los modelos de lengua y matemática muy cercana al 65%, y mantuvo la precisión de la validación cruzada en un 64% y 63% respectivamente.

Para regresión logística el Hiperparámetro que analizamos fue el parámetro de regularización C para el cual consideramos los valores 0.001, 0.01, 0.1, 1, 10, 100 obteniendo diferentes resultados en cuanto a cuál era el mejor valor a utilizar dependiendo de si consideramos lengua o matemática.

Conclusión

La precisión base de nuestros modelos se sitúa entre el 60% y el 65%. Este rendimiento, junto con el análisis de otras métricas, indica una relación significativa entre las variables seleccionadas en el dataset y los resultados finales de los estudiantes en las materias de Lengua y Matemática. Si bien es posible mejorar la precisión, consideramos que este nivel es adecuado para los fines de un sistema de alerta temprana. Dado que el objetivo es predecir si un estudiante aprobará o no, esta precisión es suficiente para motivar una intervención proactiva por parte del colegio, fomentando medidas de apoyo oportunas para los estudiantes en riesgo.

Los modelos con mejor rendimiento general fueron el bosque aleatorio y la regresión logística, ya que demostraron métricas superiores no sólo en términos de precisión base, sino también en cuanto a la curva ROC, el AUC y los resultados de validación cruzada. Estos modelos lograron un equilibrio robusto entre las distintas métricas de desempeño, destacándose sobre otros enfoques y proporcionando resultados más confiables en la predicción del desempeño académico de los estudiantes en Lengua y Matemática.

Anexos

Anexo 1: Link al collab de entrenamiento de modelos

https://colab.research.google.com/drive/1w1M3X3_Blpqqd5Gy6xBuMKoL6TeiL_rF?usp=sharing

Anexo 2: Capturas de pantalla del collab de entrenamiento de modelos

✓ Identificación de variable objetivo

En esta sección se crea un dataframe adaptado específicamente para entrenar el modelo de mpuntaje a partir del dataframe unificado. Elimino la columna lpuntaje, renombro la columna mpuntaje a "y" y la transformo en un valor binario (0 si es menor que el promedio y 1 si es mayor que el promedio).

```
[4] #Creo una copia del dataframe para trabajar en el modelo de mpuntaje
df_mpuntaje = df.copy()

#Identifico la variable como mpuntaje y la renombro como y
df_mpuntaje['y'] = df['mpuntaje'].apply(lambda x: 1 if x >= 500 else 0) #Si es modelo binario

#Borra la columna clase binaria que fue reemplazada
df_mpuntaje.drop('mpuntaje', axis=1, inplace=True)

#Eliminar la columna lpuntaje, que no será utilizada en este dataframe
df_mpuntaje.drop('lpuntaje', axis=1, inplace=True)
```

```
[6] # Mostrar cuantos valores en la columna "y" valen 0 o 1

print(df_mpuntaje['y'].value_counts())
```

```
⇒ y
0    99279
1    52346
Name: count, dtype: int64
```

```
[7] #Balancear los resultados

# Contar la cantidad de 0 y 1 en la columna 'y'
conteo_clases = df_mpuntaje['y'].value_counts()
# Calcular cuántos 0 eliminar para reducir a 52346
cantidad_a_eliminar = conteo_clases[0] - 52346

# Obtener los índices de las filas con 'y' igual a 0
indices_ceros = df_mpuntaje[df_mpuntaje['y'] == 0].index

# Seleccionar aleatoriamente los índices a eliminar
indices_a_eliminar = np.random.choice(indices_ceros, cantidad_a_eliminar, replace=False)

# Eliminar las filas seleccionadas del DataFrame
df_mpuntaje = df_mpuntaje.drop(indices_a_eliminar)

# Verificar el nuevo conteo de clases
print(df_mpuntaje["y"].value_counts())
```

```
⇒ y
1    52346
0    52346
Name: count, dtype: int64
```



```
[8] # prompt: Armar la matriz de correlación para la columna y. Eliminar todas las columnas con correlación que tenga
```

```
# Calculate the correlation matrix
correlation_matrix = df_mpuntaje.corr()

# Get the correlation with the target variable 'y'
correlation_with_y = correlation_matrix['y']

# Find columns with absolute correlation less than 0.1
columns_to_drop = correlation_with_y[abs(correlation_with_y) < 0.05].index

# Print the columns to be dropped and their correlation values
print("Columns to be dropped:")
for col in columns_to_drop:
    print(f"{col}: {correlation_with_y[col]}")

# Drop the columns from the DataFrame
df_mpuntaje = df_mpuntaje.drop(columns=columns_to_drop)
```



```
Columns to be dropped:
vive_padre: 0.04738802794908249
vive_hermano: 0.026832223214891683
tiene_tablet: 0.04482508029445094
cuida_familiar: -0.040312238922684235
realiza_tareas_hogar: 0.0006759291857699136
dias_trabajo_fuera_casa: 0.028722421452655772
fuera_escuela_leer: 0.043259533930424136
fuera_escuela_deporte: 0.03234488867198128
fuera_escuela_clases_artes: 0.048290062473476836
faltas: -0.03177040282172089
motivo_faltas_problemas_clima: -0.005785642798944287
conflicto_trado_adultos: 0.0029140041796409536
conflicto_tratado_sanciones: 0.035674166214867915
padres_extranjeros: 0.0451607434081947
```