Trabajo Practico Nº1
-Grupo 1 AA 2024-

Gastón Peña
gastonicolas1995@gmail.com

Ignacio Linari
ignaciolinari@gmail.com

Jose Chelquer
jose@chelquer.com.ar

Leonardo Iula
leonardoiula@gmail.com

Resumen

La deserción estudiantil en la educación superior es un desafío con implicaciones académicas y socioeconómicas significativas. Este trabajo explora la aplicación de modelos de aprendizaje supervisado para predecir la deserción de estudiantes de primer año, utilizando un conjunto de datos con 76,518 registros, cada uno con 43 variables demográficas y académicas. Se compararon algoritmos de clasificación binaria como la regresión logística, árbol de decisión, y máquinas de vectores soporte (y un ensamble de éstos usando *Voting*), así como modelos multiclase, incluidos KNN, árbol de decisión y ensambles. Los resultados indicaron que, aunque todos los modelos mostraron un rendimiento adecuado, los métodos de ensamble superaron ligeramente a los modelos individuales. Se concluye que las unidades aprobadas en los primeros semestres fueron las variables más predictivas.

1 Introducción

La deserción estudiantil en la educación superior constituye un problema complejo con importantes implicaciones académicas y socio-económicas. Identificar a los estudiantes en riesgo de abandonar sus estudios permite a las instituciones implementar estrategias de intervención temprana y mejorar la retención estudiantil. Este estudio se centra en el desarrollo y evaluación de modelos de clasificación para predecir la deserción estudiantil.

Utilizando un conjunto de datos que incluye variables demográficas y académicas de estudiantes de primer año, se entrenaron y compararon diversos algoritmos tanto de clasificación binaria (como la regresión logística, árboles de decisión y máquinas de vectores soporte) como multiclase (utilizando técnicas de ensamble como Random Forest y LightGBM).

El trabajo apunta no sólo a lograr buenas predicciones sino a comparar el desempeño y las particularidades de distintos modelos de aprendizaje supervisado.

2 Materiales y Métodos

2.1 Set de Datos

Se utilizó un conjunto de datos compuesto por 76.518 registros de estudiantes de primer ingreso, cada uno con 43 variables. Estas variables incluyeron información demográfica y académica recopilada al momento de la inscripción, así como el rendimiento académico durante el primer año. La variable objetivo, indicando el estado final del estudiante, se codificó en tres categorías: Desertó (25.296 casos), Graduado (36.282 casos) y En Curso (14.940 casos). Se consideró a un estudiante como "Graduado" únicamente si completó la carrera dentro del tiempo estipulado; aquellos que aún no habían finalizado

fueron clasificados como "En Curso". Los estudiantes que abandonaron formalmente sus estudios fueron categorizados como "Desertó".

Las variables utilizadas se clasificaron en tres categorías:

- Antecedentes: Edad, género, estado civil, nivel educativo del estudiante y de sus padres, y ocupación de los padres.
- Ingreso: Necesidades especiales (por ejemplo, discapacidad, movilidad geográfica), modalidad de cursado (diurno o nocturno), puntaje en el examen de ingreso, carrera elegida, edad al momento de la inscripción y variables contextuales como la tasa de inflación y desempleo del año de ingreso.
- Rendimiento académico: Número de materias cursadas, aprobadas y acreditadas en el primer año, promedio ponderado de calificaciones, y situación económica (beca, deuda, pago de matrícula).

Las variables numéricas se expresaron en escalas apropiadas (por ejemplo, calificaciones de 0 a 100, tasas de desempleo en porcentaje). Las variables categóricas se codificaron utilizando números o etiquetas textuales, según corresponda. Variables como el nivel educativo se consideraron ordinales, aunque no se aplicó una codificación específica para reflejar este orden en todos los casos

Una descripción mas detallada de los datos se encuentra en ***Anexo A***

2.1.1 Preparación de los datos

El conjunto de datos fue sometido a un proceso de limpieza y pre-procesamiento para garantizar su calidad. Se identificó un único registro con valores faltantes, el cual fue eliminado del análisis. Además, se corrigieron inconsistencias menores en la codificación de algunas variables categóricas.

Para las variables ordinales, como el nivel educativo del estudiante y sus padres, se implementó una codificación numérica que preservó el orden inherente de las categorías. La variable 'Estado civil' se transformó utilizando la técnica one-hot encoding, generando variables binarias para cada categoría.

Se evaluó la posibilidad de agrupar las categorías de otras variables, pero debido a la alta variabilidad y la falta de criterios objetivos para la agrupación, se decidió mantenerlas como variables categóricas individuales.

2.1.2 Tratamiento de desbalance de clases

Debido al desequilibrio en la distribución de las clases, se optó por un enfoque de submuestreo para mitigar el riesgo de que el modelo se sesgara hacia la clase mayoritaria. Esta técnica implica reducir el tamaño de la clase mayoritaria a través de un muestreo aleatorio, permitiendo así obtener un conjunto de datos más balanceado y mejorar la capacidad generalizadora del modelo.

Para la binarización del Target se evaluaron dos posibilidades:

1. Codificar únicamente los “Desertó” como 1, que es el target de interés, porque no hay evidencia de que los “En Curso” finalmente desertaran.
2. Codificar como 1 también a los “En Curso”, de modo que entren en el conjunto que merece atención

Optamos por la segunda opción, ya que no solo nos permite agrandar la muestra sino que nos parece interesante que este grupo es el que potencialmente necesita ayuda: algunos para no desertar, otros para terminar la carrera a tiempo. Ver tabla 1 donde también se muestra la nomenclatura para el caso multiclase.

Target	Recuento Binario	Recuento Multiclase
0	36282	14940
1	36282	14940
-1	-	14940

Table 1: Recuento Binaria: 0 (Graduado), 1 (Deserto+En Curso). Recuento Multiclase: -1 (Deserto), 0 (Graduado), 1 (En Curso

2.1.3 Pre-procesamiento de datos

Para mejorar la precisión y la interpretabilidad de los modelos, se llevó a cabo una normalización de los datos. Esta transformación es crucial en algoritmos basados en distancia, como KNN, ya que las variables con escalas diferentes pueden distorsionar la medida de similitud entre los puntos de datos. En el caso de la RL, la normalización implica alguna pérdida de legibilidad en cuanto al significado de los coeficientes, pero mejora su lectura en términos de importancia de las variables.

2.1.4 Evaluación del modelo

Se utilizó una división estratificada aleatoria para separar los datos en conjuntos de entrenamiento (70%) y evaluación (30%), con el fin de garantizar que ambos conjuntos representen adecuadamente la distribución de las clases en el conjunto de datos completo. Esta estrategia permite obtener una estimación más precisa del desempeño del modelo en datos no vistos previamente.

2.2 Metodología

Se siguió un protocolo estandarizado para el desarrollo y evaluación de los modelos:

1. *Selección y optimización de hiperparámetros:* Se identificaron los hiperparámetros más relevantes para cada modelo y se optimizaron utilizando validación cruzada estratificada.

2. *Análisis de complejidad:* Se construyeron curvas de complejidad variando un hiperparámetro clave relacionado con la complejidad del modelo, manteniendo los demás fijos en sus valores óptimos.

3. *Entrenamiento final y evaluación:* El modelo con el mejor desempeño se entrenó con el conjunto completo de entrenamiento y se evaluó en el conjunto de prueba.

4. *Análisis de importancia de variables:* Cuando fue posible, se determinó la importancia relativa de las variables predictoras.

5. *Registro y seguimiento:* Se mantuvo un registro detallado de todas las etapas del proceso, incluyendo hiperparámetros, modelos, resultados y visualizaciones.

Una lista resumida de los modelos ensayados juntos a los hiperpametros optimizados se pueden ver las Tabla 2 para clasificación binaria y para la clasificación multiclase.

Modelo	#obs / clase	Hiperparámetros optimizados y método
Regresión Logística (rl)	36,282	C
Decision Tree (dt)	36,282	Random Search: Criterion, Max_depth, min_samples_split, min_samples_leaf, ccp_alpha (todos los features)
Support Vector Machine (SVM)	36,282	C, kernel = 'linear'
Voting (voting)	36,282	Ensamble soft de los anteriores
K Neighbors (knn)	14,940	Grid: n_neighbors, weights
Decision Tree (dt_t)	14,940	Idem caso binario
Ensamble: Random Forest	14,940	Los mismos que para Decision Tree + bootstrap. Max_features: sqrt
Bagging de knn (bg_knn)	14,940	Idem knn, 10 estimadores
Ada boosting (adb)	14,940	n_estimators, learning_rate, algorithm
Boosting lightGbm (lgbm)	14,940	n_estimators, learning_rate, num_leaves, max_depth, feature_fraction

Table 2: Resumen ensayos clasificación binario y Multiclase

2.2.1 Implementación modelos

La implementación de los diferentes modelos fueron realizadas en un Jupyter Notebook en Python en el entorno de trabajo Google Colaboratory ([1]).

Se deja acceso a dicho script en el siguiente link: ***Anexo B***

En particular las implementaciones de LogisticRegression (lr), DecisionTreeClassifier (dt), SuportVectorMachine (SVM), VotingClassifier (Voting), KneighborsClassifier (knn), RandomForest (rf), AdaBoostClassifier (adb) corresponden a la libreria Scikit-learn [2] y Lightgbm (lgbm) esta basado en el trabajo descripto en [3]

3 Resultados

A continuación se presentan las tablas 3 y 4, que resumen los resultados de los hiperparámetros evaluados y las métricas de *performance* obtenidas para cada modelo. Cada modelo fue construido utilizando la configuración optimizada identificada durante el proceso de ajuste. La tabla 3 muestra los resultados para la clasificación binaria, mientras que la tabla 4 presenta los resultados para la clasificación multiclase.

En particular se trabajaron con los siguientes hiperparametros:

- **max_d**: Máxima profundidad del árbol.
- **n_est**: Número de estimadores.
- **min_ss**: Muestras mínimas para dividir un nodo.
- **min_sl**: Muestras mínimas en las hojas.
- **boot**: Bootstrap, indica si se usa remuestreo con reemplazo (True/False).
- **alg**: Algoritmo de AdaBoost.
- **lr**: Tasa de aprendizaje (learning rate).

- **feat_frac**: Fracción de características utilizadas por LightGBM.
- **leaves**: Número de hojas en el árbol de LightGBM.
- **acc_train**: Exactitud en el conjunto de entrenamiento.
- **acc**: Exactitud en el conjunto de prueba.
- **prec**: Precisión.
- **rec**: Recall.
- **f1**: Puntaje F1.

cuya implementación sera discutida en la sección de cada modelo en particular.

Modelo	**rl**	**dt**	**svm**	**Voting**
hp	C = 4.28	max_depth = 10	C = 0.06	Voting soft, rl, dt, svm
acc_train	0.881	0.888	0.881	0.886
acc	0.881	0.877	0.881	0.882
prec	0.905	0.896	0.906	0.906
rec	0.852	0.853	0.849	0.853
f1	0.878	0.874	0.877	0.879

Table 3: Resultados de los modelos binarios. Abreviatura hp para hiperparametros, acc para accuracy, prec para precision, rec para recall y f1 para la f1-score

Modelo	**knn**	**dt_t**	**rf**	**knn_t**	**adb**	**lgbm**
hp	k=30	max_d=7	n_est=310, min_ss=10, min_sl=2, max_d=20, boot=F		alg=SAMME, lr=1.0, n_est=200	feat_frac=0.5, lr=0.1, max_d=10, n_est=100, leaves=31
acc_train	1.0	0.863	0.943	0.964	0.788	0.826
acc	0.757	0.776	0.795	0.768	0.788	0.800
prec	0.769	0.781	0.802	0.776	0.795	0.806
rec	0.757	0.776	0.795	0.768	0.788	0.800
f1	0.759	0.775	0.797	0.769	0.790	0.801

Table 4: Resultados de los modelos multiclase. Abreviatura hp para hiperparametros, acc para accuracy, prec para precision, rec para recall y f1 para la f1-score

3.1 Clasificación Binaria

3.1.1 Regresión Logística

En este caso, la medida de complejidad está determinada por el hiperparámetro C, el cual es la inversa de la fuerza de regularización. Para valores bajos de C (es decir, con una regularización alta), hasta un orden de magnitud de 10^{-2}, los valores de *accuracy* son reducidos tanto en el conjunto de entrenamiento como en el de prueba. Esto sugiere que con un nivel de regularización muy fuerte, los coeficientes tienden a ser de magnitudes absolutas bajas y similares, lo que dificulta la adecuada separación de las clases. A partir de estos valores, los resultados se estabilizan.

Es importante notar que el modelo no genera coeficientes exagerados incluso cuando la regularización no se aplica, lo que permite una separación efectiva de las clases. Nuestra hipótesis es que, aunque existen variables correlacionadas, el peso de las principales variables predictoras es tan elevado (en comparación con las correlaciones con otras variables) que no se generan inestabilidades en el modelo.

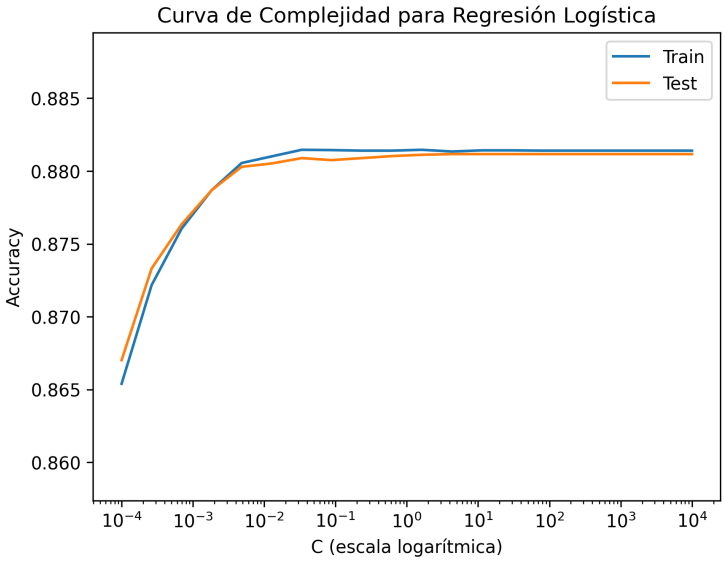

Figure 1: Curva Complejidad para Regresión Logística.

3.1.2 Árbol de Decisión

Como puede verse en la Figura 2 la precisión del conjunto de entrenamiento (curva azul) sigue aumentando a medida que se incrementa la profundidad del árbol, aunque se estabiliza después de una profundidad de aproximadamente 15. En contraste, la precisión del conjunto de prueba (curva naranja) alcanza su valor máximo cuando la profundidad está alrededor de 10, y luego comienza a decaer ligeramente para finalmente estabilizarse.

A medida que el modelo se vuelve más complejo y adapta a los datos, las curvas de entrenamiento y prueba divergen por sobreajuste: el modelo está aprendiendo patrones específicos del conjunto de entrenamiento que no generalizan bien a nuevos datos (conjunto de prueba). A profundidades más grandes, el árbol está ajustando cada vez más detalles del conjunto de entrenamiento, lo que mejora su precisión allí, pero no ayuda (e incluso puede perjudicar) la precisión en el conjunto de prueba. El hecho de que el rendimiento del modelo no mejore más allá de una profundidad de 10 sugiere que unas pocas decisiones basadas en un pequeño número de variables son suficientes para lograr una clasificación óptima. Además, el tamaño mínimo establecido para las hojas del árbol limita el crecimiento del modelo, lo que refuerza la idea de que los datos son linealmente separables y no requieren múltiples niveles de partición para una separación efectiva de las clases.

Aunque no reproducimos aquí el árbol de decisión por razones de espacio (ver ***Anexo A***), es importante destacar que se observa claramente cómo la variable principal (Unidades curriculares 2do semestre aprobadas) separa casi perfectamente los nodos, mientras que los niveles sucesivos solo refinan ligeramente los datos. Dado que hemos trabajado con datos normalizados en todos los modelos, la interpretación de los criterios de bifurcación no es completamente directa. Consideramos que sería interesante repetir este análisis utilizando las variables en su escala original, con el fin de profundizar en la interpretación del modelo.

3.1.3 Maquina de Vector de soporte

En este algoritmo evaluamos el hiperparámetro C, que controla la intensidad de la regularización. Valores bajos de C implican una mayor tolerancia a los errores de clasificación que se encuentran en el margen del hiperplano, mientras que valores altos de C imponen una mayor penalización a estos errores, lo que puede conducir a un sobreajuste (*overfitting*).

Se realizaron iteraciones sobre los valores de C en el rango [0.01, 0.03, 0.06, 0.1, 0.5, 0.7, 1, 10], encontrando que la mejor regularización se logra con $C = 0.06$, dado que este valor produjo el *accuracy* más alto en el conjunto de prueba (0.8810).

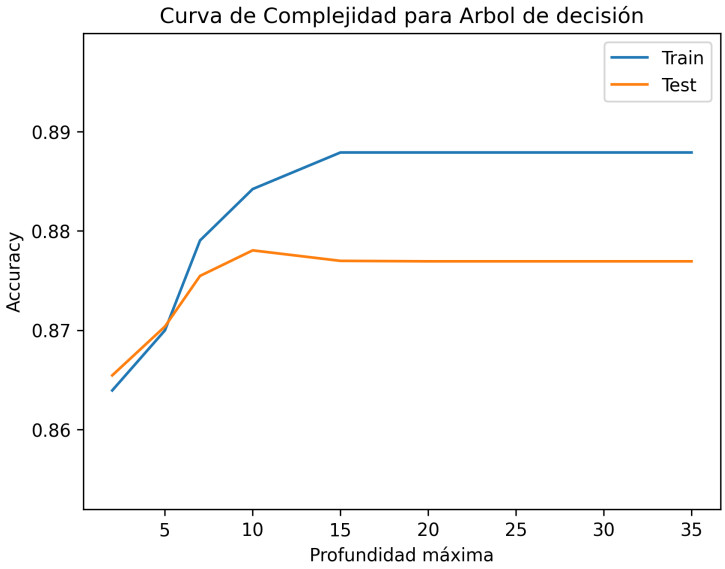

Figure 2: Curva de Complejidad para Árbol de decisión

En la curva de complejidad que obtuvimos (ver Figura 3, se observa que a medida que aumenta el valor de este hiperparámetro, también lo hace el *accuracy* en el conjunto de entrenamiento, pero a costa de una reducción en la capacidad de generalización del modelo, reflejada en una disminución del *accuracy* en el conjunto de prueba.

Utilizando un kernel lineal, es posible obtener un alto nivel de *accuracy*, comparable con los resultados obtenidos por otros modelos de clasificación. Esto sugiere que el problema de clasificación, en el enfoque adoptado, es en buena medida linealmente separable. No obstante, sería interesante explorar el uso de otros kernels para observar si, mediante una transformación de los atributos, se podrían obtener mejoras adicionales en la performance del modelo.

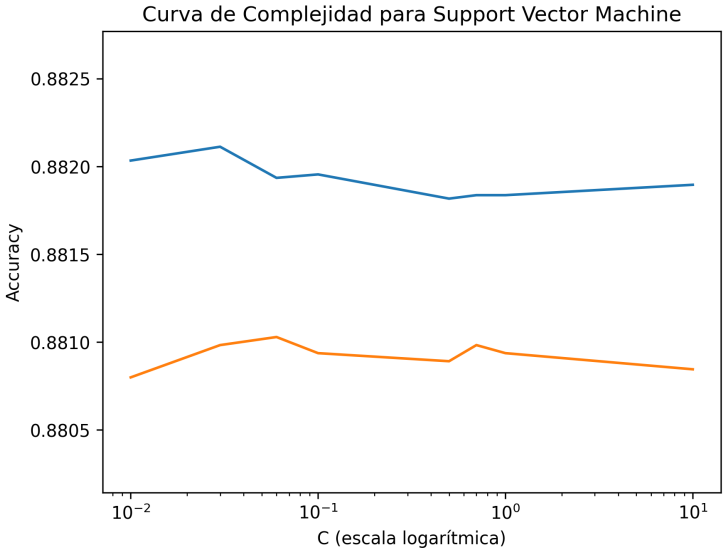

Figure 3: Curva de Complejidad para SVM

3.1.4 Ensamble Voting

El ensamble realizado con la estrategia de Voting 'soft', que incorporó a los tres modelos antes mencionados, no resultó en un incremento significativo de la *performance*, especialmente comparado contra los modelos individuales de Regresión Logística y SVM. El accuracy en test obtenido estuvo en el orden del 0.882, mientras que los modelos individuales mencionados alcazaron un 0.881 para esta medida. Lo mismo sucedió con las métricas de precision, recall y f1-score.

Entendemos que no hubo una mejora significativa en esta técnica de ensamble ya que los modelos, individualmente, no capturaron diferentes patrones en los datos para realizar sus predicciones. Un indicio de este fenómeno es el análisis de la importancia de variables, que nos indica que en los tres modelos binarios las Unidades aprobadas, ya sea en segundo o primer semestre, son las variables de mayor relevancia predictiva, ver ***Anexo A***.

3.2 Clasificación Multiclase

3.2.1 KNN

En el caso de KNN, se pueden realizar dos observaciones importantes. En primer lugar, el gráfico debe interpretarse en orden inverso, ya que a medida que disminuye la cantidad de vecinos considerados, aumenta la complejidad (o flexibilidad de adaptación) del modelo a los datos.

Las métricas tanto para el conjunto de prueba como para el conjunto de entrenamiento aumentan a medida que se reduce el número de vecinos, debido a que los vecinos más distantes no son tomados en cuenta. Sin embargo, a partir de un umbral, que para los datos de entrenamiento se encuentra en torno a los 40 vecinos, las métricas comienzan a decaer. Este resultado podría parecer sorprendente, pero se considera explicable dado que estamos utilizando todas las variables escaladas. Este proceso puede hacer que un vecino cercano comparta similitudes con el caso de análisis, posiblemente porque muchas de las variables de baja relevancia coinciden, y KNN no tiene un mecanismo para jerarquizarlas adecuadamente. Esto sugiere que para obtener resultados más precisos, es necesario consultar a un número significativo de vecinos.

Como se mencionó anteriormente, la decisión de utilizar los mismos conjuntos de observaciones y características en todos los modelos permite una comparabilidad directa, aunque penaliza a métodos como KNN. Sospechamos que una selección más cuidadosa de un número reducido de dimensiones podría mejorar los resultados, al reducirse el impacto de la maldición de la dimensionalidad.

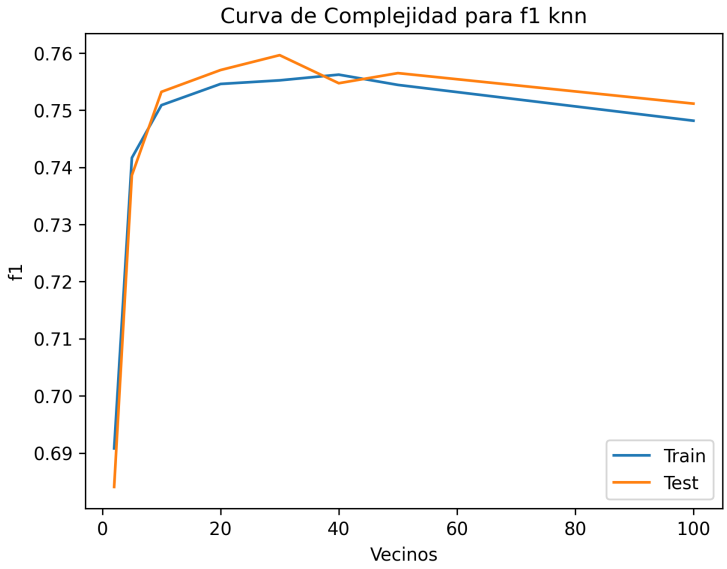

Figure 4: Curva de complejidad para F1 KNN

3.2.2 Árbol de decisión ternario

En la Figura 5 puede verse que, al igual que en el caso binario, se produce una divergencia entre las precisiones para las distintas profundidades en entrenamiento y en prueba. El valor optimo de profundidad en este caso fue de 7, donde la precisión del conjunto de prueba alcanza un máximo de alrededor de 0.78 (contra 0.87 del binario). Esto refleja que el problema ternario es más difícil, y el modelo tiene más dificultades para generalizar correctamente a nuevos datos. Con más clases, también puede haber más superposición entre éstas, lo que hace más difícil que el modelo genere reglas de decisión precisas.

Aunque no esta aquí por una cuestión de espacio, puede verse en ***Anexo A*** que se distribuye un poco más la de las variables, a diferencia del binario en donde casi toda la importancia recaía en una sola.

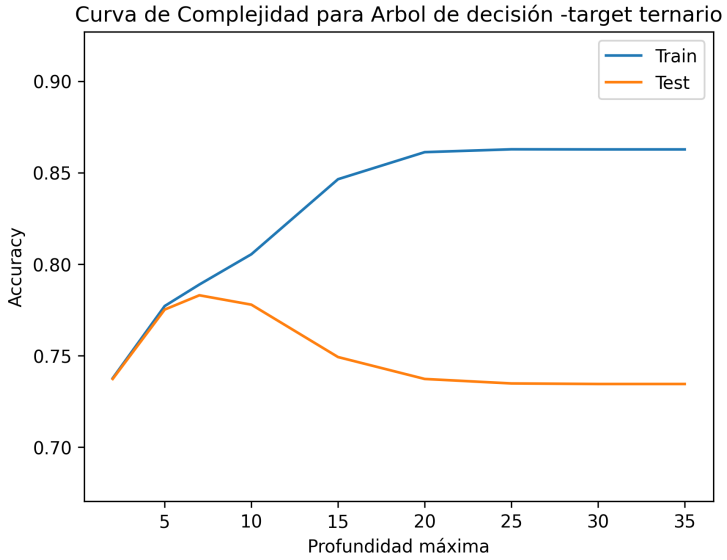

Figure 5: Curva de complejidad para Arbol de decisión ternario

3.2.3 Random Forest

Implementamos un modelo de Random Forest según lo descripto previamente en la sección de materiales y métodos (ver ***Anexo B/Bagging: Random Forest***).

Para optimizar el rendimiento del modelo, decidimos llevar a cabo un proceso de ajuste de hiperparámetros utilizando la técnica de RandomizedSearchCv, que evalúa un número fijo de combinaciones aleatorias. Esto es menos exhaustivo que un GridSearchCV, pero generalmente más rápido y puede ser más eficiente para encontrar buenos valores de los hiperparámetros dentro de espacios de búsquedas grandes, que son computacionalmente costosos.

Para evaluar el rendimiento del modelo final, se calcularon las métricas descriptas en la **Tabla 4/rf_t** tanto en el conjunto de entrenamiento como en el conjunto de prueba.

Para analizar la presencia de overfitting, se graficaron las curvas de aprendizaje, que muestran la evolución de la precisión en los conjuntos de entrenamiento y prueba en función de la complejidad del modelo (ver Figura 6)

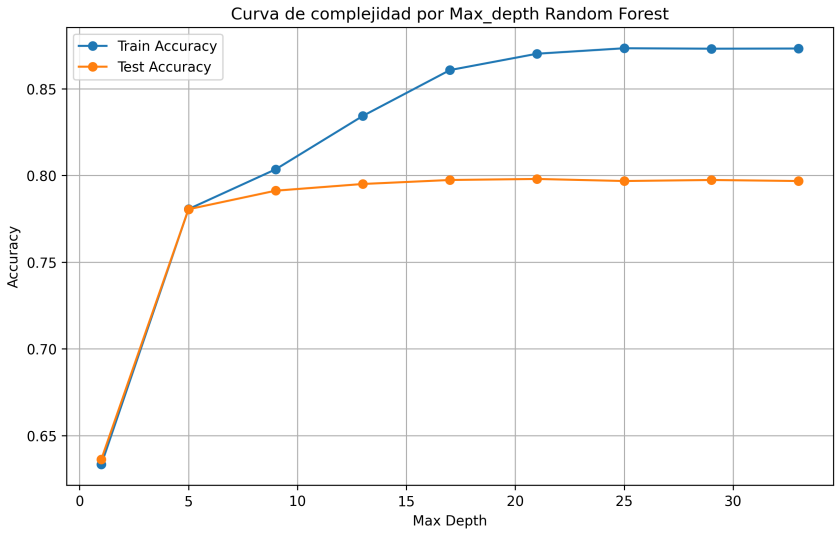

Figure 6: Curva de complejidad por Max_depth. Se muestra la precisión del modelo en los conjuntos de entrenamiento y prueba al variar el hiperparámetro max_depth. Se observa un aumento en la precisión del entrenamiento a medida que aumenta la profundidad, mientras que la precisión en el conjunto de prueba alcanza un máximo alrededor de una profundidad de 15-20, indicando un posible sobreajuste para valores mayores.

A diferencia de lo que ocurre con los árboles simples, en el caso de Random Forest, los valores óptimos se obtienen con una profundidad algo mayor (max_depth = 20), y el rendimiento en los datos de entrenamiento mejoran significativamente con profundidades aún mayores.

La selección aleatoria de características en cada árbol de un Random Forest, junto con el valor por defecto de la raíz cuadrada del número total de características, genera una diversidad de árboles con diferentes capacidades de captura de patrones. Esta diversidad permite al modelo adaptarse a complejidades locales en los datos de entrenamiento. Sin embargo, un exceso de profundidad en los árboles puede llevar a un sobreajuste, donde el modelo memoriza el ruido presente en los datos de entrenamiento en lugar de aprender patrones generalizables. Por lo tanto, se espera que exista una profundidad óptima donde el modelo logra un buen equilibrio entre ajuste y generalización. Los resultados obtenidos concuerdan con esta hipótesis, mostrando una mejora en la precisión hasta una profundidad máxima de alrededor de 20, seguida de una estabilización o incluso una disminución en la precisión del conjunto de prueba

3.2.4 Boosting

Los algoritmos de boosting han demostrado ser altamente efectivos en una amplia variedad de problemas de aprendizaje automático. En esta sección, se exploran los resultados de dos algoritmos de boosting populares: AdaBoost y LightGBM. Estos modelos fueron entrenados y evaluados según lo descripto previamente en la sección de materiales y métodos (ver ***Anexo B/Boosting***).Los resultados obtenidos permitirán comparar el rendimiento de ambos algoritmos y evaluar su capacidad para mejorar la precisión de las predicciones.

3.2.5 AdaBoost

Para evaluar el desempeño del modelo AdaBoost, se realizó una búsqueda exhaustiva de hiperparámetros (GridSearchCV) con el objetivo de encontrar la mejor configuración. Los hiperparámetros optimizados fueron:

- Algoritmo: SAMME . El parámetro algorithm especifica el método utilizado para calcular los pesos que se asignan a cada clasificador débil en cada iteración
- Tasa de aprendizaje: 1.0
- Número de estimadores: 200

Con esta configuración, se obtuvo una precisión en el conjunto de entrenamiento de 0,788 y una precisión en el conjunto de prueba de 0,788, con un valor de F1 de 0,79. Estos resultados indican un buen rendimiento del modelo, con una capacidad generalización satisfactoria.

La Figura 7 muestra las curvas de aprendizaje del modelo AdaBoost. Estas curvas se construyen al entrenar el modelo con diferentes subconjuntos del conjunto de datos y evaluar su desempeño en un conjunto de validación (crossvalidation). En esta gráfica, se observa que la precisión en el conjunto de entrenamiento aumenta a medida que se incrementa el tamaño del conjunto de entrenamiento, lo cual es esperado. Sin embargo, la precisión en el conjunto de prueba se estabiliza a partir de un determinado tamaño de conjunto de entrenamiento, indicando que el modelo no está sobreajustando significativamente a los datos de entrenamiento.

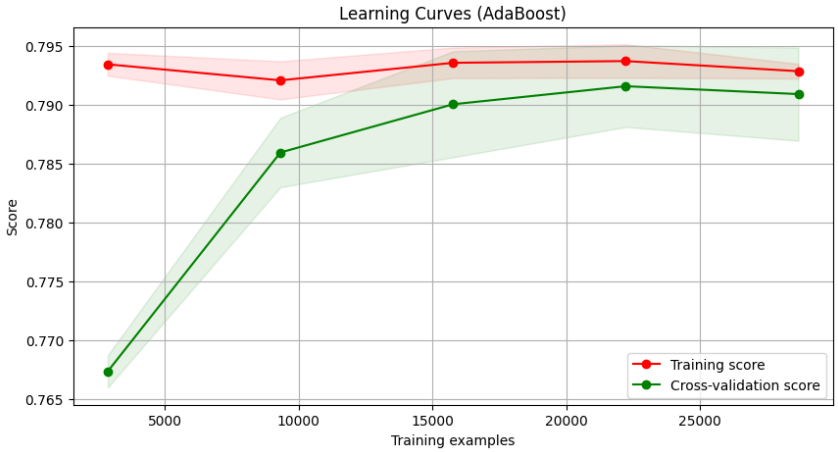

Figure 7: Curvas de aprendizaje AdaBoost.La banda de sombra alrededor de las curvas representa el intervalo de confianza de la estimación de la precisión. El hecho de que las bandas de confianza sean relativamente estrechas sugiere que las estimaciones son robustas.

3.2.6 LightGBM

Para evaluar el desempeño del modelo LightGBM, se realizó una búsqueda exhaustiva de hiperparámetros (GridSearchCV) con el objetivo de encontrar la mejor configuración. Los hiperparámetros optimizados fueron:

- feature_fraction: 0,5
- learning_rate: 0,1
- max_depth: 10
- n_estimators: 100
- num_leaves: 31

Con esta configuración, se obtuvo una precisión en el conjunto de entrenamiento de 0,826 y una precisión en el conjunto de prueba de 0,80, con un valor de F1 de 0,8. Estos resultados indican un buen rendimiento del modelo, con una capacidad de generalización satisfactoria.

La Figura 8 muestra las curvas de aprendizaje del modelo LightGBM. Estas curvas se construyen al entrenar el modelo con diferentes subconjuntos del conjunto de datos y evaluar su desempeño en un conjunto de validación (crossvalidation). En esta gráfica, se observa un comportamiento interesante: la precisión en el conjunto de entrenamiento disminuye a medida que aumenta el tamaño del conjunto de entrenamiento, mientras que la precisión en el conjunto de validación aumenta hasta cierto punto y luego se estabiliza. Este comportamiento sugiere que el modelo podría estar sobreajustando ligeramente a los datos de entrenamiento, especialmente con conjuntos de entrenamiento más grandes.

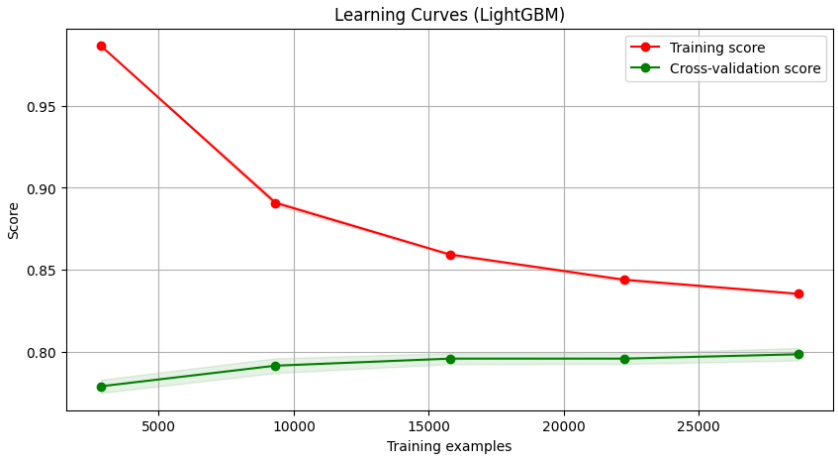

Figure 8: Curvas de aprendizaje LightGBM. La banda de sombra alrededor de las curvas representa el intervalo de confianza de la estimación de la precisión. El hecho de que las bandas de confianza sean relativamente estrechas sugiere que las estimaciones son robustas.

4 Conclusiones

Todos los modelos lograron una capacidad predictiva adecuada para la variable *Target* correspondiente. En el caso del *Target* binario, donde una clasificación aleatoria tendría una expectativa de *accuracy* del 50%, los cuatro métodos evaluados obtuvieron valores entre el 87,5% y el 88,5%. Dado que los resultados son muy similares, sería imprudente concluir que un método es superior a otro basándose en esta estrecha diferencia. Experimentos adicionales realizados fuera del alcance de este trabajo sugieren que la capacidad predictiva de algunas variables, particularmente las unidades aprobadas en el 2do semestre, es extremadamente alta. Esto explica por qué tanto los métodos basados en separación lineal (LR, SVM con kernel lineal) como los modelos basados en árboles de decisión o vecindades han logrado un rendimiento sólido.

El ensamble de *Voting Classifier* de los tres modelos binarios (Regresión Logística, Árbol de decisión y Máquinas de Vectores soporte) produjo resultados ligeramente superiores al mejor de ellos. La estrategia *Soft* de *Voting* implicó que la predicción final del ensamble promedió las probabilidades de clase de cada modelo individual, ponderadas por su precisión. En este sentido, cada modelo "votó" por la clase que consideró más probable y su voto fue pesado. Estimamos que no existió una mejora sustancial debido a que los modelos ya presentaban un alto rendimiento, por lo que su ensamble no aportó una ventaja significativa. Tal como se comentó previamente, los ensambles resultan especialmente útiles cuando los modelos individualmente ajustan predicciones sobre diferentes regiones de la estructura de datos, o bien sobre y sub estiman, de modo que al promediarse se obtengan mejores resultados. Un comportamiento similar se experimentó en la clasificación multiclase. En estos casos, los ensambles también mejoraron el rendimiento de los modelos base: RF, ADB y LGBM superaron a los modelos DT, y el *bagging* de KNN mejoró respecto a KNN. Si bien las diferencias fueron pequeñas, se concluye que en modelos con menor éxito predictivo, los ensambles podrían ofrecer mejoras más significativas, dado que las estimaciones tienden a promediarse y compensarse entre los diferentes estimadores.

Las curvas de complejidad revelaron algunos comportamientos interesantes. En la mayoría de los casos, las métricas en el conjunto de entrenamiento fueron superiores a las del conjunto de prueba. Por otra parte, se observó el patrón típico en el que la *accuracy* del entrenamiento aumenta cuando la complejidad del modelo supera el nivel óptimo, mientras que en paralelo, se produjo una disminución correspondiente en la *accuracy* del conjunto de prueba (aunque ésto no sucedió para Random Forest, en el que el *accuracy* se estabiliza).

Centrándonos en el significado de las predicciones, resulta notable que el desempeño del alumno durante el 1er año -especialmente en el 2do semestre- sea, por sí mismo, un predictor excelente. Evaluamos la posibilidad de que el "exito" de las variables de desempeño en 1er año obedeciese a que es el momento en que se decide la deserción. Los datos muestran que podría ser el caso en una

proporción importante. Sin embargo, se observa cantidad de alumnos que no tienen actividad durante el 1er año y sin embargo se gradúan. En este trabajo no codificamos la carrera de modo de incluirla en los modelos por la cantidad valores distintos de esta variable y por no disponer de un criterio claro para agruparlas. De todos modos, hicimos algunos ensayos no documentados aquí, codificando las carreras por los niveles de deserción observados (que diferen claramente entre ellas) sin que las predicciones mejoraran más que un nivel marginal. A la hora de pensar en nuevas variables que nos hubiese sido interesante incluir, nos interesaría contar con una de "nivel de satisfacción" con la carrera elegida, y con la "distancia" entre el hogar y la universidad. Finalmente, sería interesante encarar otro problema: el de pronosticar deserción al momento del ingreso. Hicimos algunas exploraciones al respecto, con resultados que no superaban claramente al azar, pero creemos que con un tratamientos mayor de las variables categóricas podría mejorarse.

Bibliografía

[1] Google. Google colaboratory. `https://colab.research.google.com/`, 2024. Accessed: 2024-10-10.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[3] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.