PROYECTO RIESGO DE INCUMPLIMIENTO CREDITICIO

Presentado por:

Valeria González González
Aura María Molina Amaya
Maryely Isabel Rubio de la Cruz

Presentado a:
Raúl Ramos Pollán

UNIVERSIDAD DE ANTIOQUIA
1 8 0 3
FACULTAD DE INGENIERÍA

UNIVERSIDAD DE ANTIOQUIA
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL
INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL PARA LAS CIENCIAS E INGENIERÍAS
2023

INTRODUCCIÓN

La Inteligencia Artificial (IA) es una de las ramas de las ciencias de la computación que más interés ha despertado en la actualidad, debido a su enorme campo de aplicación (Ponce et al., 2014). Esta herramienta utiliza una variedad de técnicas y enfoques, como el aprendizaje automático (machine learning), el cual permite crear varios modelos que analizan comportamientos para con ello generar predicciones y tomar decisiones.

El objetivo principal de este proyecto fue explorar algunos algoritmos de Machine Learning para predecir el riesgo de incumplimiento crediticio de los clientes de Home Credit, una empresa financiera no bancaria. Para abordar este problema, se utilizó un conjunto de datos proporcionado por la competencia, que incluía información diversa sobre los solicitantes de crédito. A lo largo del proyecto, se siguieron varias etapas, desde la exploración descriptiva del conjunto de datos hasta la implementación y evaluación de diferentes modelos supervisados y no supervisados

1. Planteamiento del problema

Actualmente, muchas las personas que desean adquirir nuevos préstamos no cuentan con un historial crediticio suficiente o simplemente nunca han adquirido un crédito en su vida; por lo cual deben acudir a otros medios para obtener el dinero y es allí donde prestamistas que no son acreditados para esta labor sacan provecho a beneficio propio. Por ello, y teniendo en cuenta una variedad de datos alternativos, incluida la información de telecomunicaciones y transaccional, se predecirá cuán capaz es cada solicitante de pagar un préstamo.

1.1. Dataset

Se utilizará el dataset de una competencia de Kaggle llamada **Home Credit Default Risk** (https://www.kaggle.com/competitions/home-credit-default-risk).

Para realizar el modelo, se utilizará el archivo con los datos de entrenamiento (application_train.csv), el cual cuenta con una muestra de 307.511 clientes.

Como lo que se busca predecir es el comportamiento de pago futuro de los clientes a partir de datos de comportamiento crediticio de aplicación, demográficos e históricos; en el dataset, se encuentran columnas con información del cliente como el tipo de préstamo, monto del préstamo, género, edad, si tiene carro y/o casa, número de hijos, nivel educativo, ocupación, ingresos, tipo de ingresos, entre otras, como el cumplimiento de la documentación requerida, etc.

1.2. Métricas

Como métrica se utilizará la asignada por la competencia: el área bajo la curva ROC entre la probabilidad prevista y el objetivo observado. El AUROC se calcula como el área bajo la curva ROC. Una curva ROC muestra la compensación entre la tasa positiva verdadera (TPR) y la tasa positiva falsa (FPR) en diferentes umbrales de decisión.

Una curva ROC siempre comienza en la esquina inferior izquierda, es decir, el punto (FPR = 0, TPR = 0) que corresponde a un umbral de decisión de 1 (donde cada ejemplo se clasifica como negativo, porque todas las probabilidades pronosticadas son inferiores a 1).

Una curva ROC siempre termina en la esquina superior derecha, es decir, el punto (FPR = 1, TPR = 1) que corresponde a un umbral de decisión de 0 (donde cada ejemplo se clasifica como positivo, porque todas las probabilidades pronosticadas son mayores que 0).

1.3. Variable Objetivo

Como variable objetivo, se tiene TARGET (1 - Cliente con dificultades de pago: tuvo un pago atrasado más de x días en al menos una de las primeras cuotas del préstamo en la muestra, 0 - todos los demás casos).

2. Exploración de las variables

2.1. Análisis de la variable objetivo

Como se mencionó anteriormente, nuestra variable objetivo, TARGET es una variable booleana, es decir que sólo puede tener dos tipos de entrada respecto al incumplimiento de pago del cliente, 0 (No) o 1 (sí). En la figura 1, se muestra el comportamiento de esta variable. Se aprecia que el 91.93% de los datos corresponden a clientes que sí cumplieron con el pago de un préstamo.

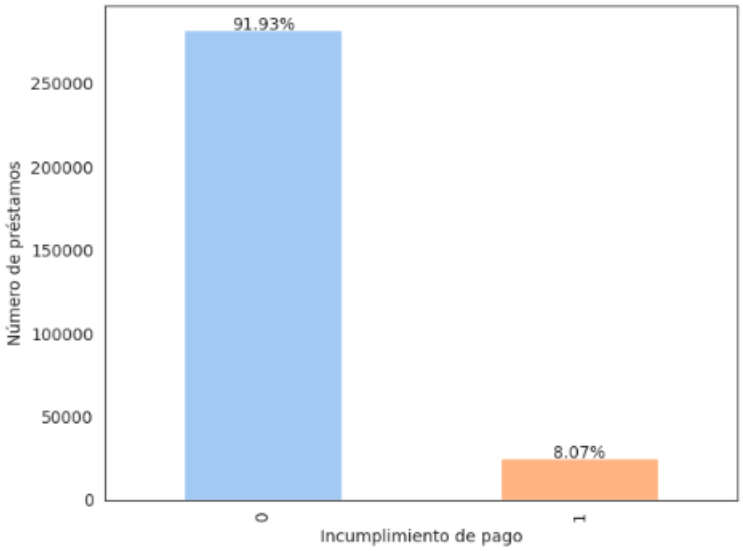

Figura 1. *Comportamiento de la variable objetivo.*

2.2. Datos faltantes

Una vez realizado el análisis de la variable objetivo, se procedió a revisar cuáles variables tienen mayor porcentaje de datos faltantes, lo cual se muestra en la tabla 1. Podemos ver que las variables con mayor número datos faltantes son *COMMONAREA_MEDI*, *COMMONAREA_AVG* y *COMMONAREA_MODE* con un 69.87% en términos porcentuales.

Tabla 1. *Variables con la mayor cantidad de datos faltantes.*

	Total	Percent
COMMONAREA_MEDI	214865	69.872297
COMMONAREA_AVG	214865	69.872297
COMMONAREA_MODE	214865	69.872297
NONLIVINGAPARTMENTS_MODE	213514	69.432963
NONLIVINGAPARTMENTS_AVG	213514	69.432963

2.3. Correlación de variables

En la tabla 2, se muestra la correlación existente entre las diferentes variables con la variable objetivo. Se puede observar que las variables *DAYS_BIRTH* y *REGION_RATING_CLIENT*, son las que mayor correlación tienen con la variable objetivo. Es de aclarar que se redujo la cantidad de variables analizadas a las 7 con mayor correlación, esto debido a la gran cantidad de las éstas que tiene la base de datos.

Tabla 2. *Correlación de siete variables con la variable objetivo.*

	TARGET
TARGET	1.000000
DAYS_BIRTH	0.078239
REGION_RATING_CLIENT_W_CITY	0.060893
REGION_RATING_CLIENT	0.058899
DAYS_LAST_PHONE_CHANGE	0.055218
DAYS_ID_PUBLISH	0.051457
REG_CITY_NOT_WORK_CITY	0.050994

2.4. Distribución de las variables numéricas

En la figura 4 se muestran las distribuciones de cada variable, aquí podemos observar que son muy pocas las variables que tienen una distribución similar a la distribución normal. También se puede observar que muchas de las variables cuentan con una distribución simétrica hacía la izquierda.

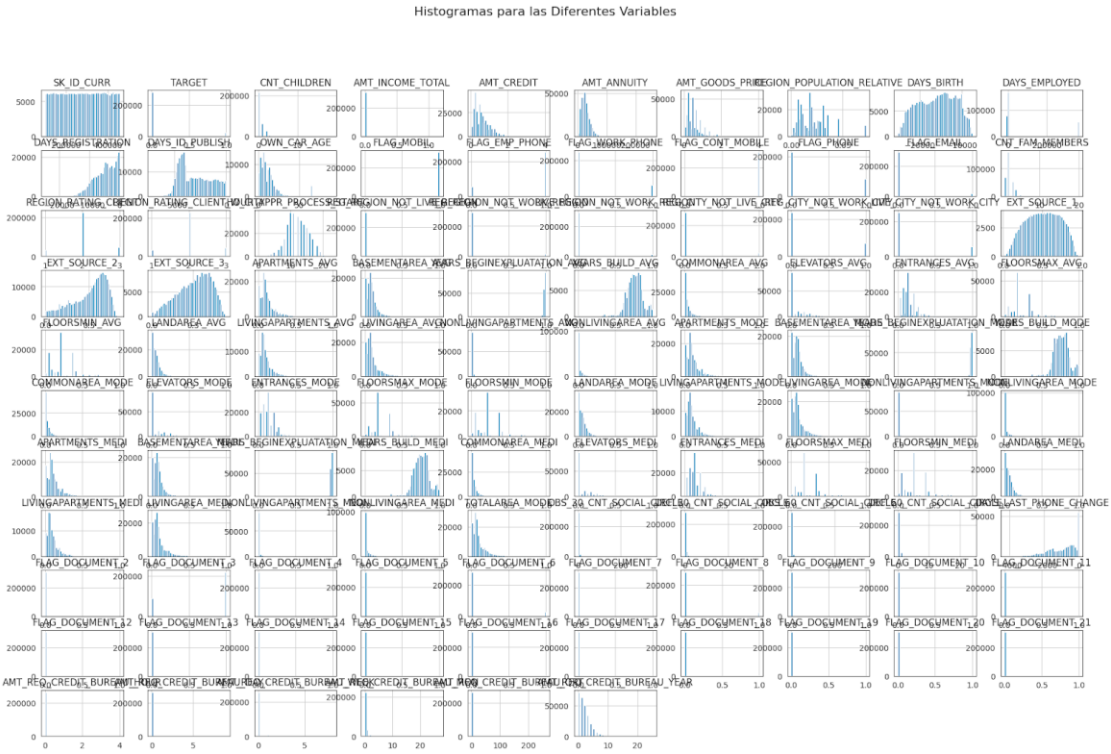

Figura 4. *Distribución de las variables.*

En la figura 5, se redujo la cantidad de gráficas de distribución a las siete seleccionadas, las cuales se escogieron las que sus columnas tuviesen mayor correlación con la variable objetivo

(mayor a 0.05). Aquí podemos observar que las variables con mayor correlación no siguen una distribución específica.

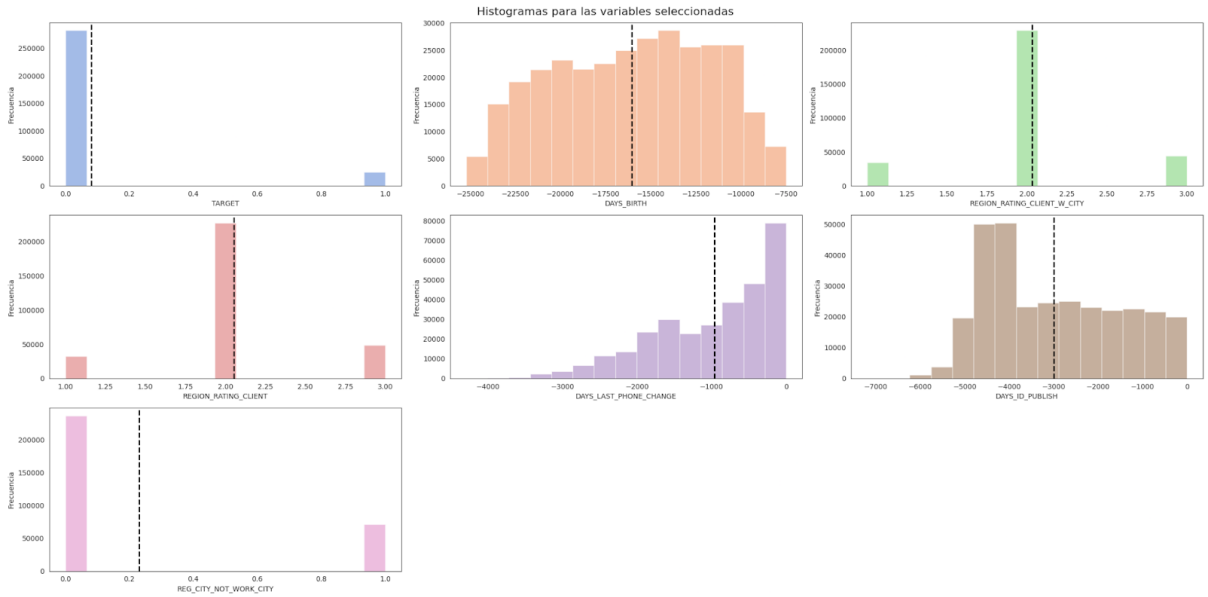

Figura 5. *Distribución de las siete variables mayor correlacionadas con la variable objetivo.*

3. Tratamiento de datos

- **Eliminación de las columnas con varios datos faltantes:** se procede a eliminar las columnas que cuentan con más del 50% de datos faltantes, ya que se considera que es mucha información la que no se tiene para poder aportar suficiente información al modelo.
- **Relleno de datos faltantes:** se hace un tratamiento del resto de los datos faltantes mediante la imputación por media para las columnas numéricas y de imputación por moda para las columnas categóricas.
- **Transformación de variables categóricas a numéricas:** se hace uso de la Codificación de etiquetas (Label Encoding) y la Codificación One-Hot (One-Hot Encoding) según el número de valores únicos que tenía cada variable categórica, **con** el fin de que que los modelos de aprendizaje automático puedan trabajar con ellas.

4. Modelos Supervisados

En este caso, se hace uso de modelos tales como Logistic Regression, Random Forest y Gradient Boosting puesto, los cuales son comúnmente utilizados para abordar problemas de clasificación binaria. Además, son modelos rápidos de entrenar y pueden producir un alto rendimiento en la mayoría de los conjuntos de datos. Se implementó la metodología de validación cruzada (cross-validation) para seleccionar el mejor modelo, en donde se obtuvo lo siguiente:

```
------------------
 Logistic Regression
test score    0.638 (±0.0589) with 10 splits
train score   0.667 (±0.0111) with 10 splits
------------------
 Random Forest
test score    0.662 (±0.0728) with 10 splits
train score   1.000 (±0.0000) with 10 splits
------------------
```

```
 Gradient Boosting
test score    0.688 (±0.0574) with 10 splits
train score   0.946 (±0.0053) with 10 splits
------------------
Selected model:
Gradient Boosting
GradientBoostingClassifier()
AUC of GradientBoostingClassifier(): 0.6733224081421424
```

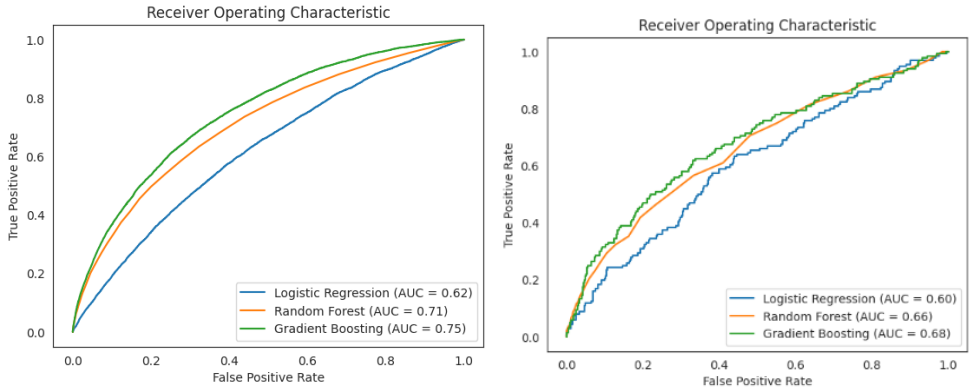

Figura 6 y 7. *Desempeño de modelos bajo el criterio de la curva ROC.*

En la Figura 6, presentamos los resultados obtenidos utilizando el conjunto de datos completo, donde se puede observar que Gradient Boosting tiene un AUC más alto en comparación con los otros dos modelos, lo que indica un mejor rendimiento en términos de capacidad de clasificación. Posteriormente, en la Figura 7, redujimos el tamaño del conjunto de datos a cinco mil registros y evaluamos nuevamente el desempeño de los modelos. Aquí también se evidencia que Gradient Boosting sigue mostrando un AUC superior a los otros modelos, reafirmando su mejor rendimiento incluso con un conjunto de datos más reducido.

4.1 Mejores hiperparámetros

Luego de evaluar el desempeño de los modelos de clasificación, se encontró que tanto Gradient Boosting como Random Forest obtuvieron resultados muy similares en términos de rendimiento. Dado esto, se decidió llevar a cabo un estudio más detallado para encontrar los mejores hiperparámetros de ambos modelos. Para ello, se utilizó una herramienta llamada GridSearchCV, que forma parte de la biblioteca de scikit-learn, la cual permite ajustar los hiperparámetros de un algoritmo específico y realizar una validación cruzada para obtener una estimación precisa del rendimiento.

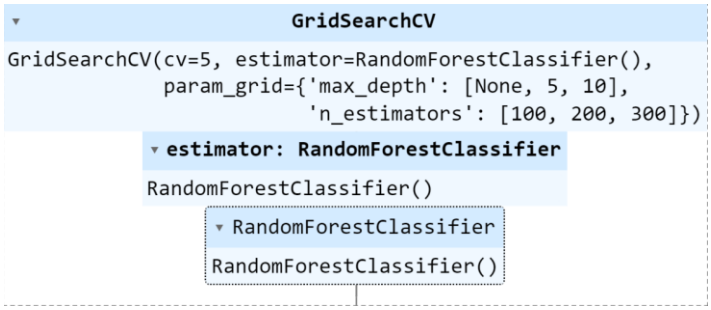

Figura 8. *Búsqueda de mejores hiperparámetros Random Forest.*

```
Best Random Forest:  RandomForestClassifier(n_estimators=300)
```

```
Best Hyperparameters for Random Forest: {'max_depth': None, 'n_estimators':
100}
Best Model AUC-ROC for Random Forest: 0.6487002975677075
```

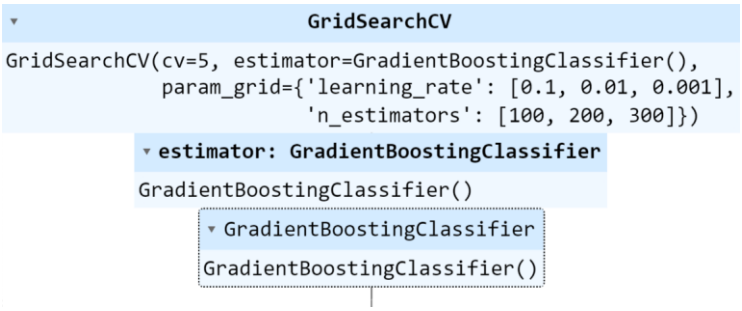

Figura 9. *Búsqueda de mejores hiperparámetros Gradient Boosting.*

```
Best Gradient Boosting:  GradientBoostingClassifier(learning_rate=0.01)
Best Hyperparameters for Gradient Boosting: {'learning_rate': 0.01,
'n_estimators': 100}
Best Model AUC-ROC for Gradient Boosting: 0.6105609582542695
```

En las Figuras 8 y 9 se muestran los valores ingresados en el módulo de Grid Search CV para Random Forest y Gradient Boosting, respectivamente. Estos valores representan los diferentes hiperparámetros que se exploraron durante el proceso de búsqueda, con el objetivo de encontrar la combinación óptima que maximice el rendimiento de cada modelo. Con el ajuste, se utilizó un valor de n_estimators=300 para Random Forest y learning_rate=0.01 para Gradient Boosting. Los resultados mostraron un valor de AUC-ROC de 0.6487 para Random Forest y 0.6106 para Gradient Boosting, lo que confirma que el modelo de Random Forest tuvo un desempeño ligeramente mejor en comparación con el modelo de Gradient Boosting.

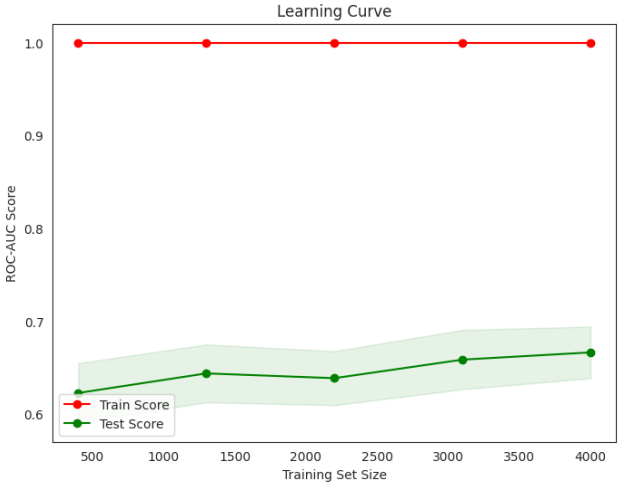

Figura 10. *Curva de aprendizaje Random Forest.*

En la figura 10, se observa la curva de aprendizaje para Random Forest con una brecha significativa entre el rendimiento de los conjuntos de entrenamiento y prueba, lo que indica la presencia de overfitting, es decir, el modelo se ajusta en exceso a los datos de entrenamiento y no generaliza a datos nuevos. Como primera instancia se recomienda controlar la complejidad del modelo o aplicar algoritmos no supervisados con el fin de reducir la dimensionalidad y mejorar la generalización,

llegado el caso que no se logre el objetivo, se puede proceder a la toma de más datos, lo cual acarrearía más costos en la elaboración del trabajo.

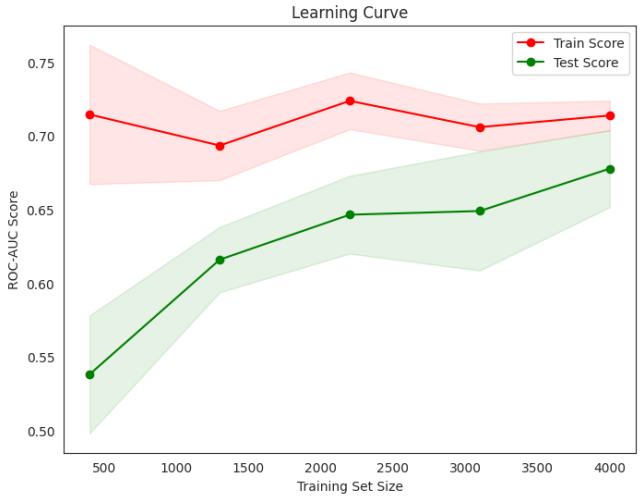

Figura 11. *Curva de aprendizaje Gradient Boosting.*

En la figura 11, se muestra la curva de aprendizaje para Gradient Boosting, donde a medida que se toman más datos para entrenar el modelo tiende a ser más estable, lo cual, en comparación con la curva de aprendizaje del modelo anterior, representa un escenario más favorable. Para este caso y debido a los resultados, una mejora que puede ser tenida en cuenta es aumentar la cantidad de descriptores, es decir las columnas, teniendo en cuenta que esto genera más inversión de tiempo y dinero, pero a la hora de presentar los resultados puede llegar a ser beneficioso teniendo en cuenta el contexto de la problemática.

5. Modelos no supervisados

Para el análisis implementando métodos no supervisados, se usaron los modelos de PCA (Análisis de Componentes Principales) y NMF (Factorización Matricial No Negativa). Estos métodos no supervisados se utilizan para reducir la dimensionalidad del conjunto de datos original. Y, aunque k-means es un algoritmo de clustering y no una técnica de reducción de dimensionalidad, puede ser utilizado como un método no supervisado para asignar etiquetas a los datos.

6. Algoritmo no Supervisado + Algoritmo Predictivo

Se llevaron a cabo 3 combinaciones de algoritmo no supervisado, como K Means o PCA, y algoritmo predictivo, como Logistic Regression o Gradient Boosting, con el objetivo de encontrar los mejores hiperparámetros para cada combinación. Se aplicó la técnica de Grid Search CV para explorar diferentes combinaciones de hiperparámetros y encontrar los valores óptimos para cada modelo. Una vez encontrados los mejores estimadores para cada combinación, se evaluó su desempeño utilizando el AUC como medida de rendimiento. Además, se generó una curva de aprendizaje para cada combinación, que muestra la evolución del rendimiento del modelo en función del tamaño del conjunto de datos de entrenamiento.

6.1. K-Means + Logistic Regression

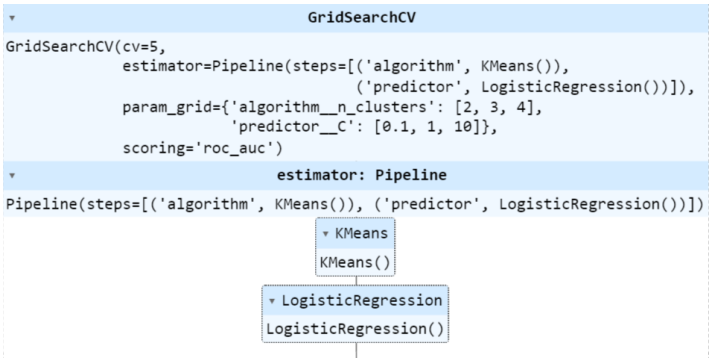

Figura 12. *GridSearchCV K-means + Logistic Regression.*

```
Best model combination 1: Pipeline (steps= [('algorithm', KMeans(n_clusters=2)),('predictor', LogisticRegression(C=10))])
Best Hyperparameters for model combination 1: {'algorithm__n_clusters': 2, 'predictor__C': 10}
AUC of Pipeline (steps= [('algorithm', KMeans(n_ers=2))clust,('predictor', LogisticRegression(C=10))]): 0.5673678195618423
```

En la figura 12, se detallan los hiperparámetros que se exploraron en el módulo de Grid Search CV para la combinación de algoritmos K-Means y Logistic Regression. Luego de su ajuste, que consiste en K-Means con 2 clusters y Logistic Regression con un parámetro C igual a 10, se observa un AUC de 0.5674, lo cual indica que el rendimiento del modelo se encuentra en un nivel intermedio, sin mostrar cambios significativos hacia una alta precisión.

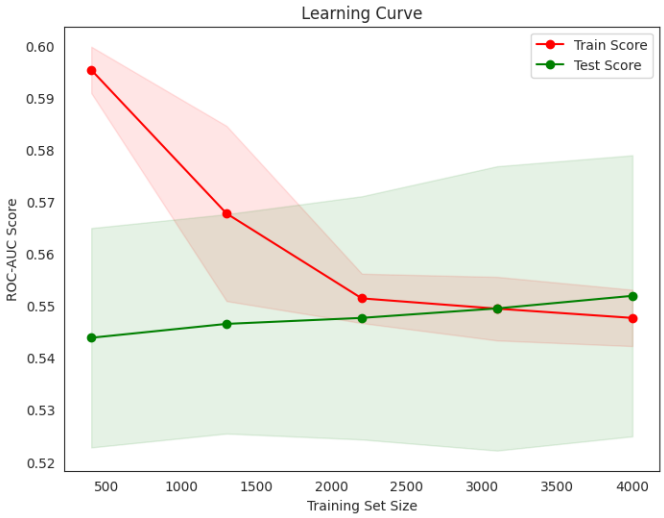

Figura 13. *Curva de aprendizaje K-means + Logistic Regression.*

En la figura 13, se muestra la curva de aprendizaje para esta combinación, en la cual se presentan bandas de confianza amplias en el conjunto de prueba y sesgo, esto puede indicar una alta variabilidad en la estimación del AUC-ROC y una mayor incertidumbre en el rendimiento del modelo. Se sugiere en primera instancia, aumentar la complejidad del modelo y en caso de no

tener los resultados esperados y realizar un análisis más detallado de puede aumentar la cantidad de descriptores es decir columnas.

6.2. PCA + Logistic Regression

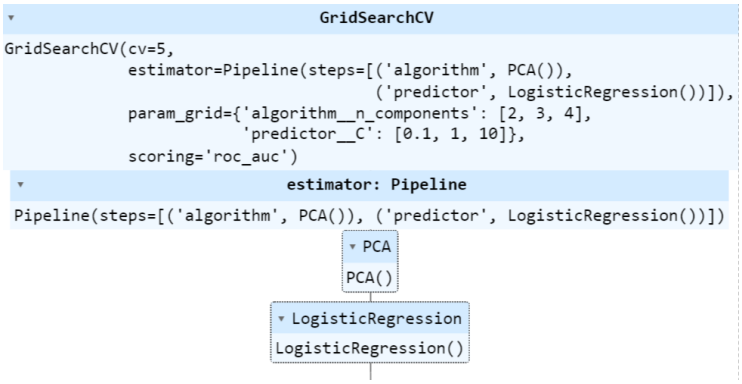

Figura 14. *GridSearchCV PCA + Logistic Regression.*

```
Best model combination 2: Pipeline (steps= [('algorithm', PCA(n_components=4)), ('predictor', LogisticRegression(C=0.1))])
Best Hyperparameters for model combination 2: {'algorithm__n_components': 4, 'predictor__C': 0.1}
AUC of Pipeline (steps= [('algorithm', PCA(n_components=4)),('predictor', LogisticRegression(C=0.1))]): 0.5753299120234604
```

En la figura 14, se presentan los hiperparámetros evaluados en el módulo de Grid Search CV para la combinación de los algoritmos PCA y Logistic Regression. El mejor modelo obtenido es PCA con 4 componentes y Logistic Regression con un valor de C igual a 0.1 arrojando un rendimiento de 0.5753, lo cual indica que el rendimiento del modelo es moderado, sin presentar mejoras significativas en comparación con las combinaciones anteriores.

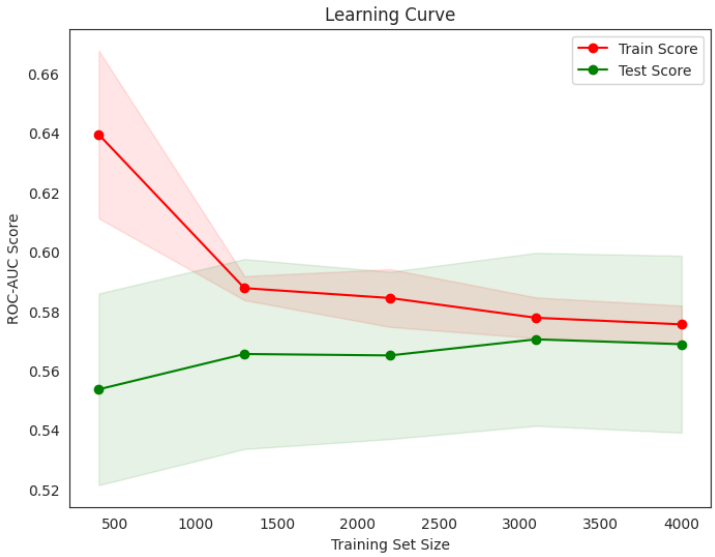

Figura 15. *Curva de aprendizaje PCA + Logistic Regression.*

En la figura 15, se presenta un sesgo en la curva de aprendizaje y una alta variabilidad en el rendimiento del modelo en diferentes conjuntos de prueba, esto se puede deber a que los datos están muy mezclados o el modelo es muy sencillo; para ello, hay dos soluciones, la primera es crear

un modelo más complejo y la segunda solución, con la cual se debe asumir costos adicionales, es añadir más descriptores, es decir columnas a la base de datos.

6.3. KMeans + Gradient Boosting

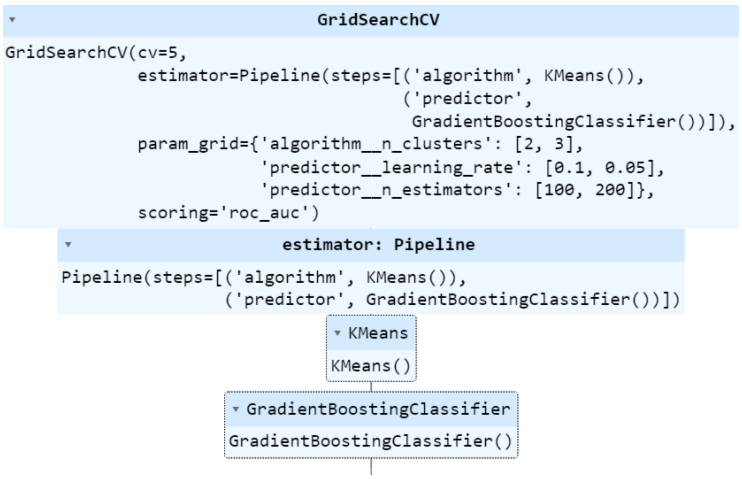

Figura 16. *GridSearchCV K-means + Gradient Boosting.*

```
Best model combination 3: Pipeline (steps= [('algorithm', KMeans(n_clusters=2)), ('predictor', GradientBoostingClassifier(n_estimators=200))])
Best Hyperparameters for model combination 2: {'algorithm__n_clusters': 2, 'predictor__learning_rate': 0.1, 'predictor__n_estimators': 200}
AUC of Pipeline (steps= [('algorithm', KMeans(n_clusters=2)), ('predictor', GradientBoostingClassifier(n_estimators=200))]): 0.5682411160945317
```

En la Figura 16 se muestran los hiperparámetros evaluados en el módulo de Grid Search CV para la combinación de los algoritmos K-Means y Gradient Boosting. Se encontró que la mejor configuración de hiperparámetros para esta combinación fue la siguiente: KMeans con 2 clusters y Gradient Boosting con una tasa de aprendizaje (learning rate) de 0.1 y 200 estimadores (n_estimators). El AUC obtenido para esta combinación fue de 0.5682, lo cual indica el rendimiento del modelo en términos de la curva ROC.

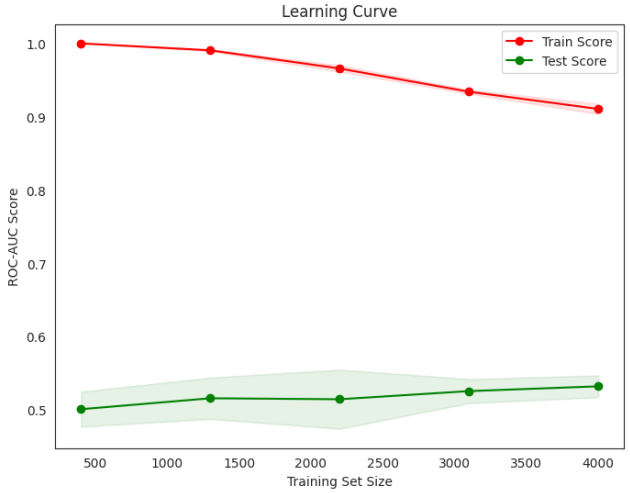

Figura 17. *Curva de aprendizaje K-means + Gradient Boosting.*

De la figura 17 se puede decir que al llevar a cabo la curva de aprendizaje de K-means + Gradient Boosting, esta presentó un overfitting, ya que hay una brecha significativa entre los datos entrenamiento y validación, lo cual indica que el modelo está sobreajustando los datos de entrenamiento, lo que puede ayudar a concluir que quizás el modelo solo está memorizado lo datos y no está aprendiendo sus patrones generales. Inicialmente, se recomienda reducir la complejidad del modelo, ya que parece estar memorizando los ejemplos individuales en lugar de aprender patrones generales. Si este paso no logra mejorar la curva de aprendizaje se recomienda aumentar la cantidad de datos, teniendo en cuenta que esta sería la última opción por los costos asociados a la misma.

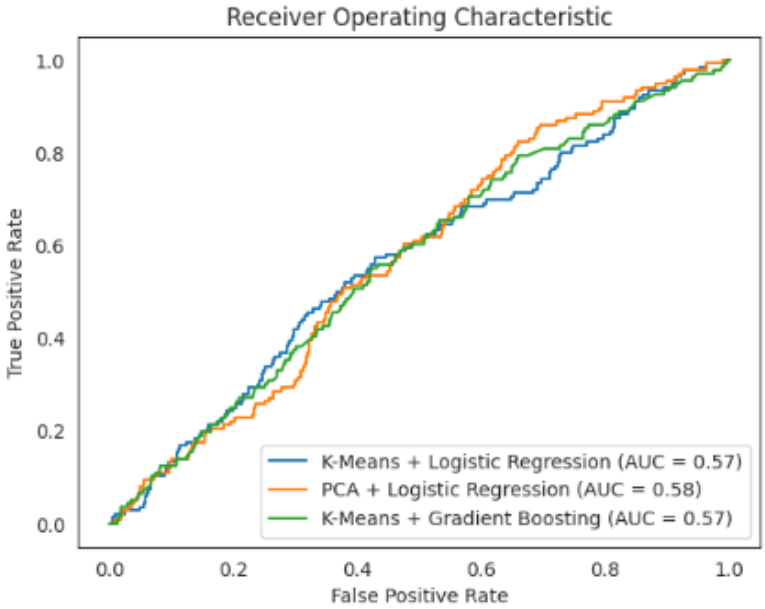

Figura 18. *Desempeño de modelos bajo el criterio de la curva ROC.*

En la Figura 18 se muestra el análisis de los gráficos de curva ROC y los valores AUC para tres combinaciones de modelos: K-Means seguido de Logistic Regression con un AUC de 0.57, PCA seguido de Logistic Regression con un AUC de 0.58, y K Means seguido de Gradient Boosting con un AUC de 0.57. La curva ROC es una representación visual de cómo el modelo realiza la clasificación, y el AUC es una medida del rendimiento del modelo, donde valores más cercanos a 1 indican un mejor rendimiento en la clasificación. En este caso, los tres modelos muestran un rendimiento similar, con valores de AUC alrededor de 0.57-0.58. Esto significa que los modelos son capaces de distinguir entre las clases de manera moderada, pero aún hay margen para mejorar el rendimiento. Adicionalmente, si se comparan estos desempeños con los de los modelos supervisados, los cuales tuvieron resultados entre 0.62-0.75, se puede observar que desmejoraron.

7. Retos y Consideraciones.

Desplegar un modelo en producción presenta desafíos y condiciones cruciales. Para establecer el nivel de desempeño mínimo, se deben definir métricas de evaluación y umbrales aceptables basados en los requisitos del negocio. El despliegue implica configurar una infraestructura adecuada y desarrollar una integración eficiente del modelo en el entorno de producción. Los procesos de monitoreo del desempeño deben incluir la captura de métricas en tiempo real, la detección de anomalías y la recopilación de retroalimentación para ajustes y actualizaciones. Un enfoque multidisciplinario y el cumplimiento de las mejores prácticas de seguridad y privacidad son fundamentales para garantizar un despliegue exitoso y un rendimiento óptimo del modelo.

8. Conclusiones

Tras todos los análisis realizados podemos concluir que:

- Es necesario analizar de forma más rigurosa cuáles son las variables que a la hora de entrenar los modelos tiene mayor relevancia para las predicciones de este, ya que con esto podemos reducir sobreajuste y errores inherentes a la ejecución de los distintos algoritmos.

- Al llevar a cabo el análisis de los algoritmos no supervisado y algoritmos predictivos, los modelos no lograron mostrar un desempeño significativo, ya que sus AUC se mantuvieron en un rango medio, lo que nos muestra que se debe considerar otras alternativas para que el modelo logre tener un aprendizaje exitoso y logre capturar los patrones subyacentes en cada uno de ellos.

- La mayoría de los casos en donde se midió el desempeño de los modelos se presentaron problemas de overfitting, lo cual se puede solucionar reduciendo la complejidad de estos modelos, ya que esta solución en primera instancia no requiere de costos adicionales.

Referencias bibliográficas

Draelos, R (s.f). Measuring Performance: AUC (AUROC) https://glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc/

Kaggle. (s.f.). Home Credit Default Risk. https://www.kaggle.com/competitions/home-credit-default-risk/overview

Moyete. (2022, 8 marzo). Curso scikit-learn | Curva ROC | Machine Learning Python 05 [Vídeo]. YouTube. https://www.youtube.com/watch?v=RZiWJIyaQbg

Ponce Gallegos, J. C., Torres Soto, A., Quezada Aguilera, F. S., Silva Sprock, A., Martínez Flor, E. U., Casali, A., . . . Pedreño, O. (2014). Inteligencia Artificial. doi:10.13140/2.1.3720.0960