

Modelo De Machine Learning Dentro De Un Enfoque De Inteligencia Artificial Para El Otorgamiento De Créditos Financieros De Libre Inversión En El Área De Crédito Social En La Caja De Compensación Familiar Comfenalco Cartagena

Stevens R. Bohorquez Ruiz

Universidad Autónoma de Occidente, Cali, Colombia

stevensrbr@gmail.com

Resumen - Este estudio explora la aplicación de inteligencia artificial, específicamente machine learning, en la optimización del proceso de solicitudes de crédito en el área de Crédito Social de Comfenalco Cartagena. Se identifican los principales desafíos actuales, como la subjetividad en la evaluación de solicitudes, la demora en los procesos y el riesgo financiero. La metodología empleada incluye el análisis de los procedimientos existentes y la propuesta de un modelo basado en machine learning para la automatización y mejora de la toma de decisiones crediticias. Los resultados esperados apuntan a una reducción en los tiempos de aprobación, una mayor precisión en la evaluación de riesgos y una alineación con estándares de calidad (ISO 9001, 14001, 45001, 27001, 42000 y 30301) y con los ODS Objetivos de Desarrollo Sostenible.

Índice de Términos - Aprendizaje Automático, Automatización de Procesos, Banca, Crédito Social, Despliegue de Modelos ML, EDA Análisis Exploratorio de Datos, Entrenamiento de Modelos, Evaluación Crediticia, Finanzas, Fintech, Gestión de Riesgos, Inclusión Financiera, Inteligencia Artificial, ISO 9001 – 14001 – 45001 – 27001 - 42000 – 30301, Machine Learning, Optimización Financiera, Toma de Decisiones, Transformación Digital, Workflow Machine Learning.

1. INTRODUCCION

1.1 Contexto

La inteligencia artificial (IA) ha revolucionado múltiples sectores mediante la automatización de procesos y la optimización de la toma de decisiones. En particular, el uso de machine learning que ha demostrado ser una herramienta clave en la industria financiera, permitiendo mejorar la precisión en la evaluación crediticia y reduciendo riesgos asociados al otorgamiento de préstamos. A nivel global, instituciones financieras han implementado modelos de IA para analizar grandes volúmenes de datos, identificar patrones y minimizar la morosidad en los créditos.

En Colombia, las cajas de compensación familiar desempeñan un papel crucial en la mejora de la calidad de vida de los trabajadores y sus familias, ofreciendo beneficios que incluyen subsidios, educación, vivienda y crédito social. Comfenalco Cartagena, como una de estas entidades, proporciona soluciones financieras mediante su área de Crédito Social, que gestiona una variedad de productos crediticios con un enfoque

en poblaciones vulnerables y afiliados. Sin embargo, los procesos actuales de evaluación y aprobación de créditos son manuales, lo que genera demoras, subjetividad en las decisiones y una mayor tasa de riesgo crediticio. [1]

1.2 Problema

El proceso de solicitud y aprobación de créditos en Comfenalco Cartagena es altamente manual y demanda una considerable inversión de tiempo por parte de los analistas. Actualmente, los funcionarios deben revisar solicitudes, analizar información financiera, validar documentos y tomar decisiones basadas en criterios que pueden verse influenciados por sesgos humanos. Este esquema genera varios problemas:

- Altos tiempos de respuesta: La evaluación manual retrasa la aprobación de créditos, afectando la satisfacción de los afiliados.
- Subjetividad en las decisiones: La falta de un modelo automatizado puede resultar en criterios inconsistentes en la aprobación o denegación de créditos.
- Incremento de la tasa de morosidad: La evaluación manual puede llevar a errores en la asignación de créditos, lo que aumenta el riesgo financiero.
- Pérdida de eficiencia operativa: Los analistas de crédito invierten tiempo en tareas repetitivas que podrían ser optimizadas mediante tecnologías de IA.

1.3 Objetivos

1.3.1 Objetivo General

Implementar un modelo de machine learning para optimizar el análisis y otorgación de créditos en el área de Crédito Social de Comfenalco Cartagena, reduciendo tiempos de respuesta y aumentando la precisión en la asignación de créditos.

1.3.2 Objetivos Específicos

- Automatizar el proceso de análisis de riesgo crediticio para reducir tiempos de evaluación.
- Mejorar la precisión de los estudios crediticios mediante análisis de datos.
- Disminuir la tasa de mora y aumentar la sostenibilidad financiera del programa de crédito social.
- Mejorar la experiencia del usuario reduciendo tiempos de espera y trámites.

2. METODOLOGÍA

En proyectos de investigación y/o desarrollo de entrenamientos de modelos de machine learning se debe hacer de manera previa un análisis exploratorio de datos (EDA), para llevar a cabo esto se realizaron unos pasos, etapas y procesos, que se detallan a continuación:

Lenguaje de programación utilizado: Python versión 3.9.5 [2]
 Entorno de desarrollo integrado IDE: VS Code versión 1.97.2 [3]

Tecnología para ejecutar fragmentos de código: Jupyter Notebook [4]



Fig. 1. Workflow Machine Learning.

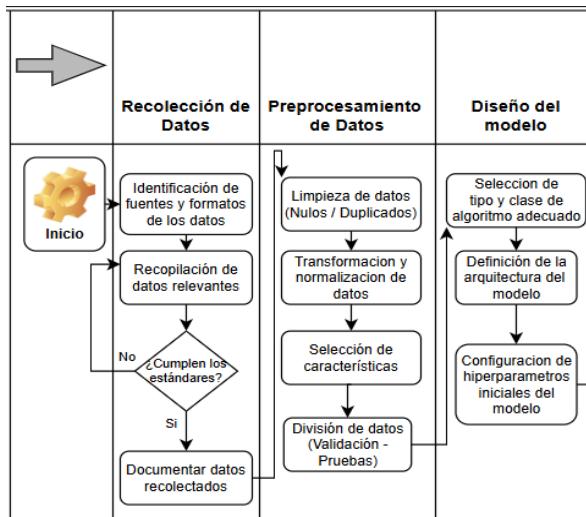


Fig. 2. Parte a. Diagrama Esquemático.

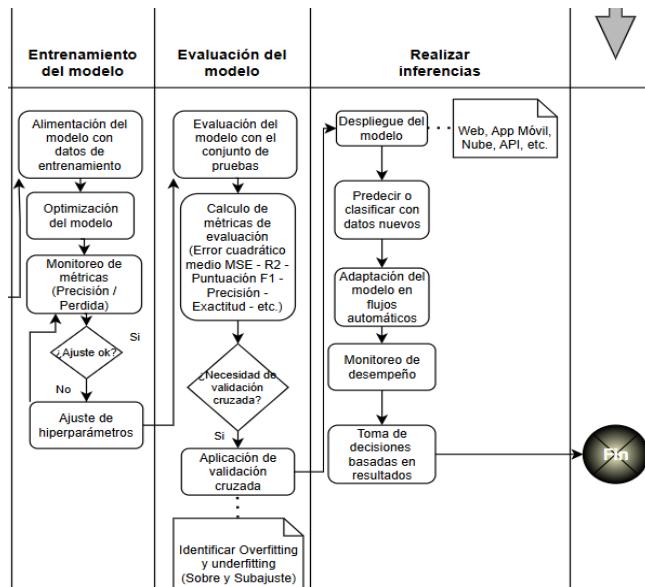


Fig. 2. Parte b. Diagrama Esquemático.

2.1 Recolección de datos

Los datos reales recolectados fueron de una base de datos empresarial real y en producción de la Caja de compensación Comfenalco Cartagena, pero no se contó con el tiempo y medios suficientes para recibir con antelación el consentimiento por parte de las personas afiliadas dueñas de los datos para el estudio.

Al existir diversas restricciones para usar datos personales como la ley de protección de datos 1581 de 2012 y el decreto 1074 de 2015 (protección de datos personales), ISO 27001 de seguridad de la Información (gestión de privacidad) y falta de un documento de consentimiento explícito de autorización; no se desistió en este estudio investigativo y se optó por utilizar un dataset con conjunto de datos públicos de Kaggle, con variables y cantidades de registros similares a los pensados en utilizar antes, de esta manera con los resultados futuros solo se deberían seguir los pasos, procedimientos o metodología utilizada en este proyecto para la solución a los retos en la Caja de compensación Comfenalco.

El dataset es accedido mediante este link: https://www.kaggle.com/datasets/nikhil1e9/loan-default?utm_source=chatgpt.com y con información relacionada en este otro link <https://github.com/alexandaniel-23/Loan-Default-Prediction-Dataset>, extraídos del “desafío de predicción de impago de préstamos” de Coursera, con un conjunto de datos de 255347 filas u observaciones y 18 columnas o descripciones en total. [5]

Descripción De Las Columnas:

- LoanID: Identificador único para cada préstamo
- Age: Edad
- Income: Ingresos anuales del prestamista
- LoanAmount: Cantidad de dinero prestado
- CreditScore: Puntaje crediticio
- MonthsEmployed: Meses empleado
- NumCreditLines: Cantidad de líneas de crédito
- InterestRate: Interés impuesto
- LoanTerm: Duración del préstamo en meses
- DTIRatio: Relación deuda-ingresos
- Education: Nivel educativo
- EmploymentType: Tipo de contrato
- MaritalStatus: Estado civil
- HasMortgage: Tiene una hipoteca
- HasDependents: Tiene dependientes
- LoanPurpose: Propósito del préstamo
- HasCoSigner: Tiene un cofirmante
- Default (Target): Variable objetivo que indica si el préstamo incumplió

2.2 Preprocesamiento y limpieza de datos:

El tipo de problema posible sería de clasificación con (Regresión Logística Binaria), porque la regresión logística es muy útil para los casos en los que se desea predecir la presencia o ausencia de una característica en una variable dependiente, clase o target y como se observa que los únicos dos valores presentes son (0 para quienes si cumplen) y (1 para quienes no cumplen) nos lleva a analizar que no sería multivariable y si es binaria, además que los valores son discretos y no continuos y así pueden ser linealmente separables.

Se utilizan las siguientes librerías para realizar el EDA:

Pandas, Numpy, Seaborn, Matplotlib, Sklearn y se importan los archivos para aplicar Perceptron, LogisticRegression, RandomForestClassifier en pruebas manuales, así como también archivos de confusion_matrix, classification_report para desplegar métricas y LabelEncoder para preprocesamientos, transformación, escalado y normalización en casos de ser requerido.

Después de importado el Dataset a un Dataframe, se renombran las etiquetas quedando de esta manera:

- ID_Prestamo
- Edad
- Ingresos_Anuales
- Monto_Prestamo
- Puntaje_Credito
- Meses_Empleado
- Cantidad_Lineaas_Credito
- Tasa_Interes_Impuesto
- Duracion_Prestamo_Meses
- Relacion_Deuda_Ingresos
- Nivel_Educacion
- Estado_Laboral
- Estado_Civil
- Tiene_Hipoteca
- Tiene_Dependientes
- Proposito_Prestamo
- Tiene_CoFirmante
- Cumplimiento

Se revisan las primeras filas del dataset, se verifican los registros de cantidad de filas (255347) y cantidad de columnas (18), los tipos de datos para verificar que valores son categóricos, numéricos, decimales, int, float, texto o cadena, object y demás; se verifican los campos nulos, NaN o None y faltantes para tomar decisiones como sacarlos del estudio de acuerdo a sus relevancias o importancias o rellenarlos con ceros o con datos estadísticos como la media o los datos más frecuentes, también se identifica si existen filas duplicadas.

Se analizan las características categóricas (Nivel_Educacion, Estado_Laboral, Estado_Civil, Tiene_Hipoteca,

Tiene_Dependientes, Proposito_Prestamo, Tiene_CoFirmante) resultando 7 columnas si no tenemos en cuenta el ID_Prestamo porque es un valor único.

Se aplica una distinción por cada una de estas características para identificar sus divisiones o contenidos.

2.2.1 Análisis exploratorio de datos (EDA):

Se generan visualizaciones y mediciones de estadística descriptiva para identificar medidas patrones o distribuciones, identificar las variables claves que podrían influir mayormente en el comportamiento del modelo, así como “Insights” preliminares describiendo las observaciones relevantes obtenidas.

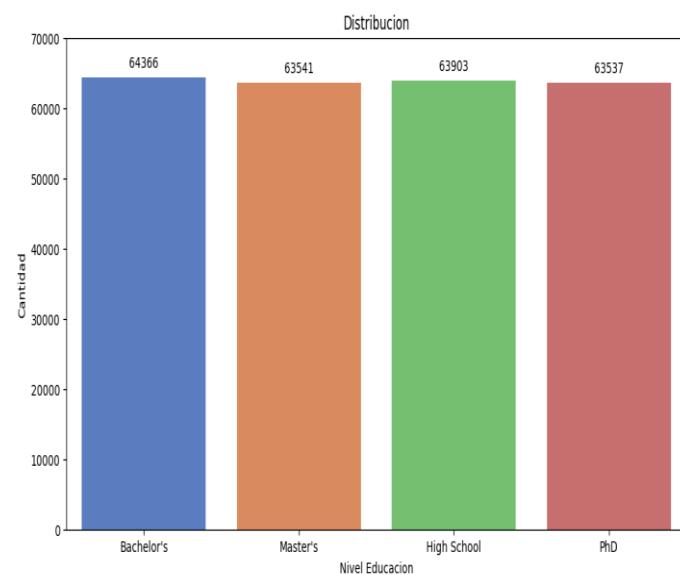


Fig. 3. Distribución por Nivel de Educación.

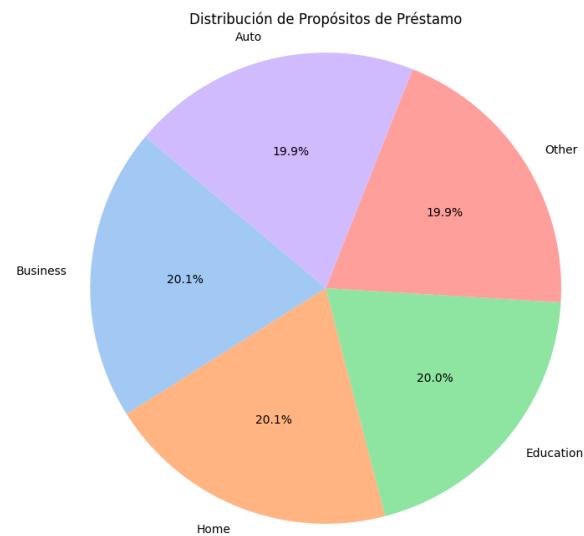


Fig. 4. Distribución por Propósito del Préstamo

2.3 Diseño del modelo:

Se generan graficas de caja o (boxplot) para analizar posibles presencias de valores atípicos (Outliers) y con esto tener un indicio de posibles tendencias a (overfitting o underfitting) posteriores y saber manejarlos en el entrenamiento.

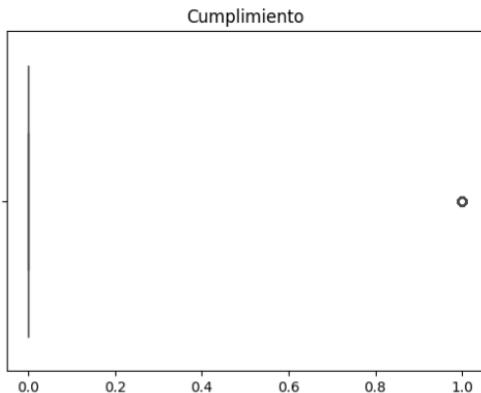


Fig. 5. Grafica de caja.

Comprobamos que el tipo de problema es de clasificación con (Regresión Logística Binaria), porque solo se observan dos únicos valores presentes en la variable objetivo (0 para quienes si cumplen) y (1 para quienes no cumplen).

Se genera un mapa de calor con el fin de mirar la correlación existente entre las características e identificar cuales están fuertemente relacionadas positiva y negativamente, y cuales no se relacionan.

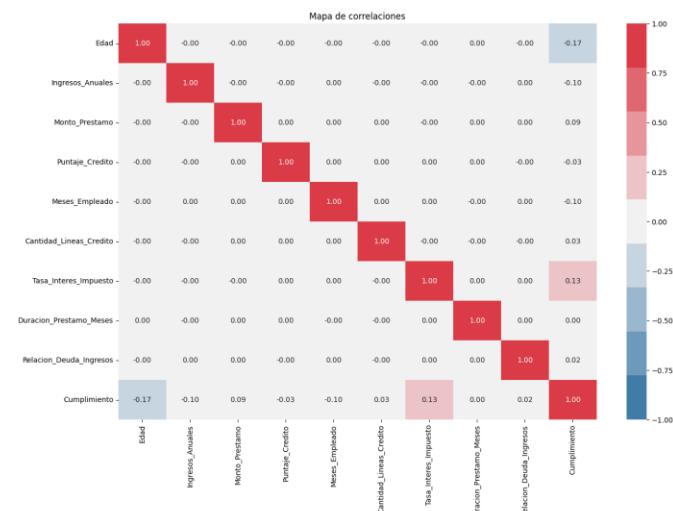


Fig. 6. Mapa de calor.

Se genera un gráfico de pastel y de barras de la variable objetivo para ver la relación en porcentajes en los dos únicos valores presentes:

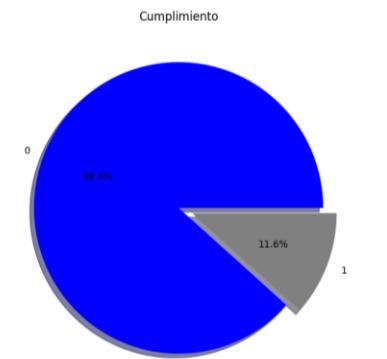


Fig. 7. Parte a. Grafica de pastel Vs Grafica de barras.

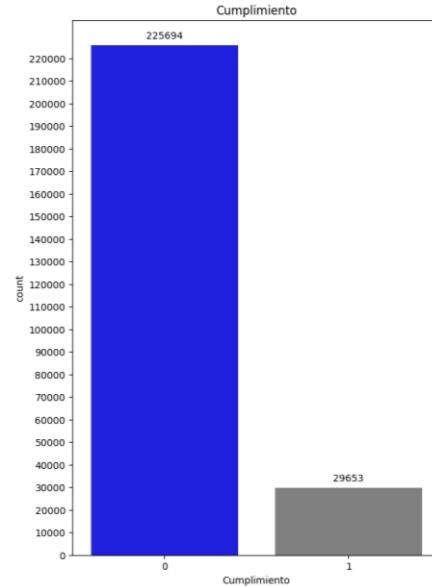


Fig. 7. Parte b. Grafica de pastel Vs Grafica de barras.

2.4 Entrenamiento del modelo

División del conjunto de datos: Se separan los datos en conjuntos de un 70% para entrenamiento y 30% para pruebas, además en el conjunto de entrenamiento se excluye la columna ID_Prestamo porque no aporta valor al estudio y se aplica la técnica (OneHotEncoder) para transformar las características categóricas en un formato numérico.

Elección de algoritmos y técnicas de aprendizaje automático:

Primero se aplican las técnicas de árbol de decisión (Decisión Tree Classifier), Bosques Aleatorios - (Random Forest Classifier) y KNNeighbours (Vecinos más Cercanos KNN) porque son algoritmos que trabajan muy bien con este tipo de problemas, pero inicialmente se aplican sin PCA (Análisis de Componentes Principales) a los datos para validar resultados.

Métricas de evaluación:

Decisión Tree Classifier: Predice mucho la clase 0 y casi nada la clase 1, tiene un Accuracy engañoso, tiene Desbalanceo porque presenta más muestras de una clase que de la otra.

Random Forest Classifier: Logra mejor precisión (88.47%), pero el desempeño en la clase 1 es de cuidado porque el Recall es muy bajo (0.03) entonces casi no está identificando los casos positivos.

KNNeighbours: Parametrizado con 6 vecinos, el modelo esta sesgado hacia la clase mayoritaria (0) tiene problema de balance de clases, su Accuracy es del 88% pero para la clase 1 (minoría) el modelo tiene precision 39% y con el Recall de solo 3% se supone que está fallando en los casos positivos.

Primera conclusión o análisis:

El modelo que mejor se ajusta es Random Forest, pero se debe mejorar posiblemente aplicando PCA

Aplicando PCA (Análisis de Componentes Principales):

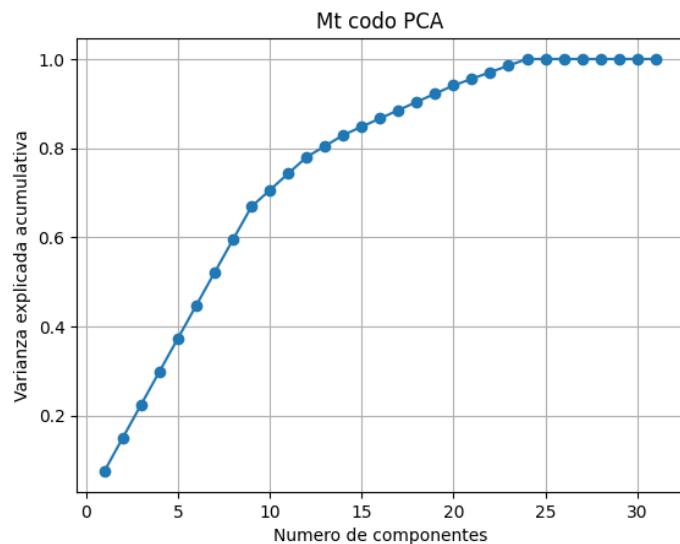


Fig. 8. Gráfico (Mt codo PCA) para ver la varianza explicada acumulativa.

Se genera un gráfico (Mt codo PCA) para ver la varianza explicada acumulativa y después se toma la decisión de trabajar con 20 el número de componentes para tratar de evitar el overfitting, intentando mejorar el rendimiento de los modelos utilizados antes, pero dejando espacio a la generalidad y al mismo tiempo sin perder mucha información, nos ubicamos donde la curva empieza a nivelarse.

Decisión Tree Classifier (Con PCA): Para la clase 0 predijo de forma correcta todo, pero para la clase 1 no predijo, con un Accuracy de 0.88.

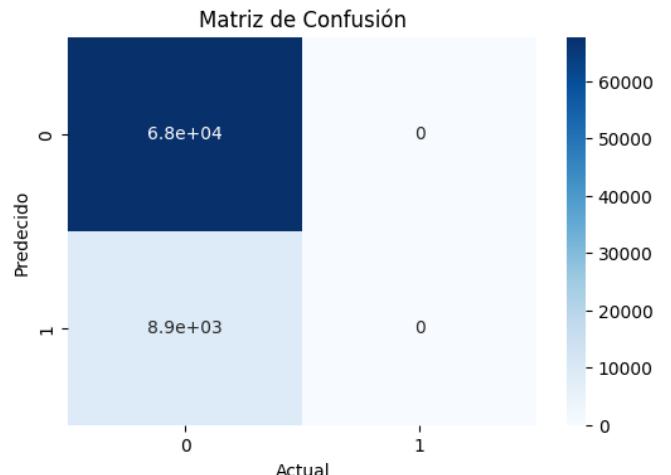


Fig. 9. Matriz de confusión Decisión Tree Classifier (Con PCA).

Random Forest Classifier (Con PCA): Buen rendimiento con la clase 0 pero existe dificultad en el modelo para identificar la clase 1.

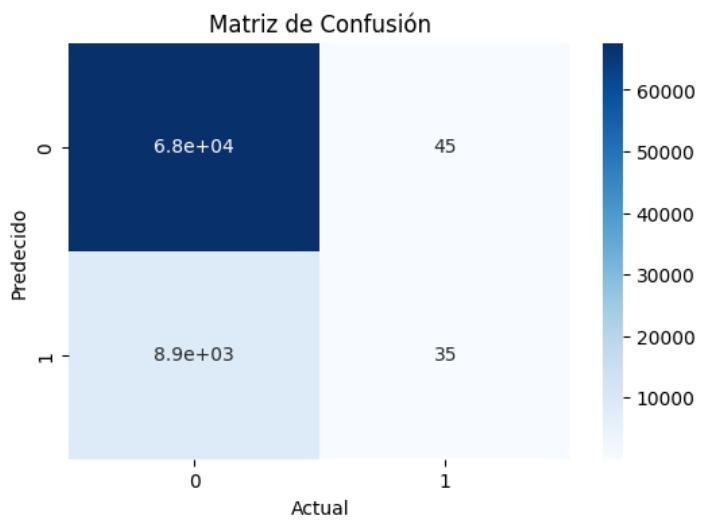


Fig. 10. Matriz de confusión Random Forest Classifier (Con PCA)

Segunda conclusión o análisis:

Decisión Tree Classifier (Con PCA): Tiene una diferencia entre la precision de entrenamiento y prueba pequeña (0.09%) esto sugiere que el modelo no está sobre ajustado y generaliza bien.

Random Forest Classifier (Con PCA): Hay un margen considerable de diferencia entre la precision de entrenamiento y prueba (11.68%) lo cual indica que el modelo está sobre ajustado (overfitting), entonces modelo se ajusta muy bien a los datos de entrenamiento, pero no generaliza bien con datos nuevos.

Resultados e inferencias iniciales:

comparación final: En este caso el modelo de árbol de decisión parece ser más adecuado porque generaliza mejor, pero aun así se debe buscar otra técnica y aplicar otros algoritmos para mejorar y afinar el modelo final.

Para buscar mejorías en el modelo final se usa la librería de Python PyCaret para analizar mejor el comportamiento de este dataset en otros algoritmos diferentes a los ejecutados hasta ahora. Al mismo tiempo se aplica la técnica de (Get_dummies) que es una función de Python para convertir variables categóricas en variables ficticias y la técnica SMOTE (Synthetic Minority Over-sampling Technique) para generar instancias sintéticas o artificiales para equilibrar la muestra de datos basado en la regla del vecino más cercano, teniendo en cuenta que el dataset tiene muchos registros y la clase (1 – No Cumple) no es muy representativa con el 11.6% lo cual afecta el entrenamiento; todo esto se hace para preparar los datos “antes” del análisis y ejecución con PyCaret y se le da la indicación para que active el uso de la GPU local probando también con modelos o algoritmos más avanzados o profundos.

Model		Accuracy	AUC
lightgbm	Light Gradient Boosting Machine	0.8770	0.9151
xgboost	Extreme Gradient Boosting	0.8719	0.9049
rf	Random Forest Classifier	0.8639	0.9093
et	Extra Trees Classifier	0.8569	0.9096
catboost	CatBoost Classifier	0.8556	0.8947
gbc	Gradient Boosting Classifier	0.8549	0.8922
ada	Ada Boost Classifier	0.8362	0.8710
dt	Decision Tree Classifier	0.7929	0.7737
qda	Quadratic Discriminant Analysis	0.7487	0.7977
nb	Naive Bayes	0.7376	0.7844
knn	K Neighbors Classifier	0.7305	0.7820
lda	Linear Discriminant Analysis	0.7160	0.7544
ridge	Ridge Classifier	0.7152	0.7544
lr	Logistic Regression	0.7118	0.7486
dummy	Dummy Classifier	0.6667	0.5000
svm	SVM - Linear Kernel	0.6057	0.6265

Fig. 11. Parte a. Resultado de recomendaciones para posterior entrenamiento manual.

Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0.6632	0.9538	0.7824	0.7005	0.7230	3.3900
0.6581	0.9395	0.7740	0.6884	0.7097	0.6540
0.6639	0.9017	0.7648	0.6721	0.6877	8.9430
0.6822	0.8596	0.7607	0.6605	0.6695	4.0910
0.6132	0.9298	0.7390	0.6450	0.6717	7.0300
0.5953	0.9513	0.7323	0.6399	0.6730	40.8100
0.6274	0.8411	0.7186	0.6065	0.6195	9.1350
0.7158	0.6799	0.6974	0.5402	0.5406	2.2470
0.5168	0.6563	0.5783	0.4029	0.4087	0.2330
0.4797	0.6424	0.5492	0.3696	0.3774	0.2130
0.6246	0.5905	0.6071	0.4022	0.4026	1.1760
0.4083	0.6106	0.4993	0.3032	0.3150	0.2640
0.3819	0.6177	0.4719	0.2915	0.3073	0.2070
0.3692	0.6122	0.4605	0.2800	0.2967	8.9670
0.0000	0.0000	0.0000	0.0000	0.0000	0.1530
0.4279	0.4628	0.3425	0.1144	0.1430	14.2420

Fig. 11. Parte b. Resultado de recomendaciones para posterior entrenamiento manual.

2.5 Evaluación del modelo:

La recomendación más factible que nos arroja PyCaret es Light Gradient Boosting Machine (LightGBM) basada en árboles de decisión, con estas métricas (**Accuracy** 0.8770, **AUC** 0.9151, **Recall** 0.6632, **Prec** 0.9538, **F1** 0.7824, **Kappa** 0.7005, **MCC** 0.7230).

Se procede a realizar el nuevo entrenamiento manual con LightGBM y se obtienen los siguientes resultados:

(**Accuracy** 0.8770, **AUC** 0.9151, **Recall** 0.6632, **Prec** 0.9538, **F1** 0.7824, **Kappa** 0.7005, **MCC** 0.7230).

Verificando de esta manera la veracidad del resultado para poder proceder con el afinamiento del modelo y tratar de subir el Recall el cual aún sigue muy bajo.

2.5.1 Fine-Tuning

Fine-Tuning con RandomizedSearchCV:

Se utilizo esta técnica (RandomizedSearchCV) porque hace la validación de forma cruzada en búsqueda aleatoria para encontrar los mejores hiperparametros sin mucho consumo computacional, pero indicando que la métrica a mejorar debe ser al Recall.

Manualmente se ingresan los hiperparametros de entrada a ajustar:

```
'num_leaves': [31, 50, 70],
'learning_rate': [0.01, 0.05, 0.1],
'n_estimators': [100, 200, 500],
'min_child_samples': [10, 20, 30],
'feature_fraction': [0.8, 1.0],
'Lambda_11': [0.0, 0.1, 0.5],
'Lambda_12': [0.0, 0.1, 0.5]
```

2.6 Inferencias

Se obtienen los mejores hiperparametros: {'num_leaves': 70, 'n_estimators': 500, 'min_child_samples': 10, 'learning_rate': 0.05, 'lambda_12': 0.0, 'lambda_11': 0.5, 'feature_fraction': 1.0} y con esto se crea el nuevo modelo para entrenarlo con los datos de entrenamiento y después hacer predicciones, donde se obtienen los nuevos resultados o las nuevas métricas manuales donde se evidencian las mejoras:

Nuevas Métricas

Accuracy: 0.9078

Precision: 0.9747

Recall: 0.7413

F1 Score: 0.8421

AUC-ROC: 0.9314

Antes del ajuste Después del ajuste Mejora		
Accuracy	0,8770	0,9080
AUC	0,9151	0,9316
Recall	0,6632	0,7418

Tabla 1. Comparación de métricas.

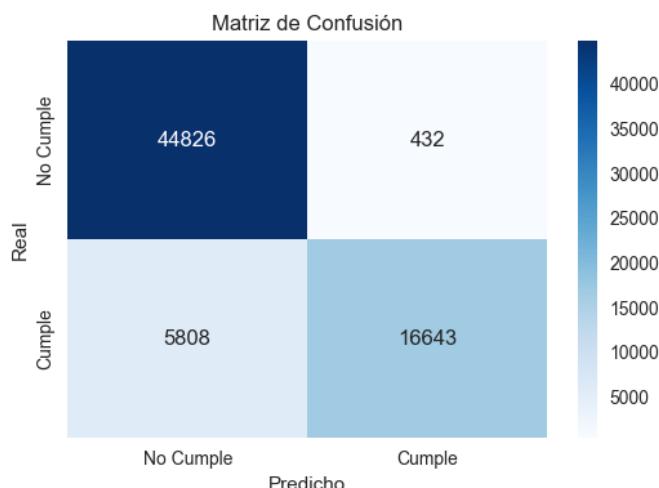


Fig. 12. Matriz de confusión del modelo afinado.

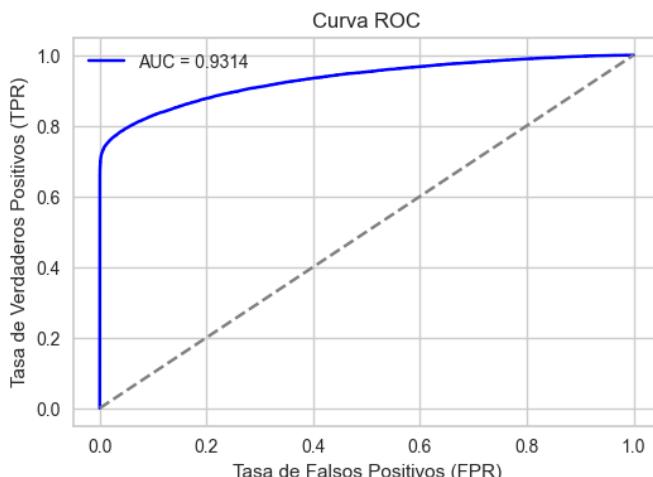


Fig. 13. Curva ROC del modelo afinado.

El ajuste de hiperparametros mejoró todas las métricas clave:

Precisión aumentó de 0.9538 → 0.9746 → Mayor confianza en predicciones positivas.

Recall pasó de 0.6632 → 0.7418 → Ahora detectamos más casos positivos.

F1-Score subió de 0.7824 → 0.8424 → Balance entre precisión y recuperación mejoró.

Kappa & MCC aumentaron → Mejor discriminación entre clases.

Conclusión: Se logró un mejor equilibrio entre precisión y recuperación, lo que indica que el modelo ahora generaliza mejor sin sacrificar la detección de casos positivos.

2.6.1 Despliegue del modelo

El modelo entrenado con **Light Gradient Boosting Machine (LightGBM)** se guarda en un archivo de Python bajo la estructura Joblib, con el estado de sus cálculos y poder reanudar su ejecución en otros entornos o máquinas.

Se desarrolla una aplicación software de escritorio utilizando el lenguaje de programación Python y la librería TKinter y como motor de base de datos se utiliza SQL Server, después se hace uso del modelo guardado desde la aplicación; finalmente se crea un ejecutable y una clave de licencia con los cuales se crea un instalador para sistemas operativos como Windows, de esta manera se finaliza con el último paso del workflow del proyecto de machine learning y se logra que las personas o clientes puedan instalar, configurar y solucionar la problemática tratada con ese producto de software.

REFERENCIAS

- [1] Comfenalco Cartagena, «Comfenalco Cartagena,» Comfenalco Cartagena, 01 01 2024. [En línea]. Available: <https://www.comfenalco.com/>. [Último acceso: 01 03 2025].
- [2] Python Software Foundation, «<https://www.python.org/downloads/release/python-395/>,» python.org, 03 05 2021. [En línea]. Available: <https://www.python.org/downloads/release/python-395/>. [Último acceso: 01 03 2025].
- [3] visualstudio, 03 02 2022. [En línea]. Available: https://code.visualstudio.com/updates/v1_98. [Último acceso: 01 03 2025].
- [4] <https://jupyter.org/>, «<https://jupyter.org/>,» jupyter.org, 01 05 1999. [En línea]. Available: <https://jupyter.org/>.
- [5] <https://www.kaggle.com/>, «<https://www.kaggle.com/>,» <https://www.kaggle.com/>, 01 11 2020. [En línea]. Available: <https://www.kaggle.com/>. [Último acceso: 01 03 2025].