Avance 5. Modelo final

A01150742 José Ovalle Alvarado

A01794879 Carlos de Jesús Méndez Tornero

A01104775 Alejandro Munguía Salazar

Proyecto Integrador | Fecha: 1/06/2025

# Introducción.

En este avance el enfoque se centra en el desarrollo e implementación de modelos de ensamble que integren los mejores modelos individuales obtenidos previamente para mejorar significativamente la calidad de las predicciones y su capacidad de generalización sobre datos no vistos así mismo, en un primer vistazo a la forma de implementación, predicciones a nuevas observaciones . El ensamble de modelos es una técnica ampliamente reconocida en el aprendizaje automático, donde la combinación estratégica de varios clasificadores puede producir un modelo con desempeño superior al de cualquiera de sus componentes por separado.

Aplicaremos diversas estrategias de ensamble, tanto homogéneas que combinan múltiples instancias de un mismo tipo de modelo, como heterogéneas que combinan diferentes algoritmos. Además, se optimizarán hiperparámetros de los modelos más relevantes para maximizar su rendimiento y se implementarán técnicas de stacking y blending, utilizando los modelos individuales mejor evaluados como base.

# Antecedentes.

El proyecto tiene como objetivo principal optimizar la gestión de morosidad en condominios residenciales mediante el uso de técnicas de inteligencia artificial. En etapas previas, hemos realizado un análisis exploratorio de datos, desarrollado ingeniería de características y construidos modelos individuales baselines que permitieron establecer un punto de partida para la predicción de morosidad.

El análisis reveló un conjunto de variables relevantes que explican el comportamiento de pago de los residentes, así como un desequilibrio significativo en la variable objetivo que fue tratado con técnicas de balanceo como SMOTE.

En avances anteriores se construyeron modelos individuales utilizando regresión logística, Random Forest, SVM, MLP y XGBoost, con resultados prometedores. Sin embargo, se identificó la necesidad de mejorar el rendimiento y la robustez general del sistema predictivo mediante técnicas de ensamble, que permiten aprovechar las fortalezas y reducir las debilidades de cada modelo individual.

# **Preparación de Datos y Modelos Base**

El conjunto de datos utilizado fue sometido a un exhaustivo preprocesamiento para asegurar su idoneidad en el modelado. Este proceso incluyó la imputación de valores faltantes mediante técnicas adecuadas al tipo de variable, garantizando que no se perdiera información crítica. Para las variables categóricas, se aplicaron métodos de codificación como la codificación ordinal, en casos donde existía un orden intrínseco, y one-hot encoding para aquellas sin jerarquía definida. Además, se discretizaron algunas variables continuas cuando se consideró beneficioso para el modelo, y se normalizaron todas las características utilizando StandardScaler para homogeneizar su escala y evitar sesgos en el aprendizaje.

Los modelos base seleccionados fueron XGBoost y MLP (Multi-Layer Perceptron), debido a su demostrado rendimiento en fases anteriores. Estos modelos fueron optimizados mediante GridSearchCV, una técnica que permite explorar sistemáticamente combinaciones de hiperparámetros para encontrar la configuración que maximiza su eficacia.

Gráfico, Gráfico de rectángulos

El contenido generado por IA puede ser incorrecto.

Gráfico

El contenido generado por IA puede ser incorrecto.

# **Construcción de Modelos de Ensamble**

Para mejorar la precisión y robustez del modelo final se reescribieron los mejores modelos MLP y XGB en forma de pipeline de SKL y luego, se implementaron diversas estrategias de ensamble, cada una con sus ventajas específicas:

* Bagging (Bootstrap Aggregating): Esta técnica se aplicó utilizando múltiples instancias de Random Forest, cada una entrenada con un subconjunto aleatorio de los datos generado mediante muestreo con reemplazo. Las predicciones de estos modelos se combinaron mediante votación mayoritaria, lo que redujo la varianza y mejoró la generalización.
* Boosting: Se empleó el algoritmo XGBoost, que construye modelos de forma secuencial, corrigiendo en cada iteración los errores cometidos por los modelos anteriores. Este enfoque permitió mejorar significativamente la capacidad predictiva, especialmente en la identificación de patrones complejos en los datos.
* Stacking: En esta estrategia, se combinaron las predicciones de los mejores modelos individuales (XGBoost, MLP, Logistic Regression y SVM) mediante un meta-modelo. Este último, implementado como una regresión logística, aprendió a ponderar las contribuciones de cada modelo base, aprovechando así sus fortalezas individuales.
* Blending: Similar al stacking, pero con la particularidad de utilizar un conjunto de validación independiente para entrenar el meta-modelo. Esto ayudó a prevenir el sobreajuste y aseguró que el modelo final mantuviera un buen desempeño en datos no vistos.

Todas estas técnicas se implementaron utilizando librerías especializadas como scikit-learn y XGBoost, así como herramientas personalizadas para el stacking y blending.

Texto

El contenido generado por IA puede ser incorrecto.

# **Evaluación Comparativa de Modelos**

La evaluación de los modelos se realizó mediante validación cruzada estratificada y sobre un conjunto de prueba independiente. Se utilizaron métricas clave como el F1-score, que equilibra precisión y recall, resultando especialmente útil en problemas con clases desbalanceadas. Además, se calculó el accuracy como medida general de rendimiento, aunque con la precaución de interpretarlo en contexto debido al desbalance de clases. El ROC AUC proporcionó información sobre la capacidad discriminativa del modelo, mientras que el tiempo de entrenamiento permitió evaluar su eficiencia computacional.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modelo | F1 | Accuracy | ROC\_AUC | Tiempo |
| Stacking | 0.848924 | 0.869710 | 0.954030 | 71.717085 |
| XGBoost | 0.833728 | 0.882843 | 0.953589 | - |
| Voting | 0.831754 | 0.881593 | 0.953288 | 10.192000 |
| MLP | 0.828893 | 0.879508 | 0.951683 | - |

# **Selección del Modelo Final**

Para la elección del modelo final se consideró no solo su rendimiento en términos de métricas, sino también su eficiencia computacional y alineación con los objetivos del negocio. Se priorizó la minimización de falsos negativos, crucial para detectar casos de morosidad de manera oportuna, y se buscó mantener un equilibrio entre precisión e interpretabilidad, asegurando que el modelo pudiera ser entendido y utilizado en producción.

Gráfico, Gráfico de rectángulos

El contenido generado por IA puede ser incorrecto.

Gráfico, Gráfico de rectángulos

El contenido generado por IA puede ser incorrecto.

# **Visualización e Interpretación del Modelo Final**

Para el modelo seleccionado, se generaron visualizaciones clave que permitieron comprender su comportamiento y resultados:

* Curva ROC: Mostró la relación entre la tasa de verdaderos positivos y falsos positivos, destacando la capacidad del modelo para distinguir entre clases.
* Matriz de Confusión: Proporcionó un desglose detallado de los aciertos y errores, facilitando la identificación de áreas de mejora.
* Curva de Precisión-Recall: Ayudó a evaluar el rendimiento en condiciones de desbalance, mostrando el trade-off entre precisión y recall para diferentes umbrales.
* Importancia de Características: Reveló las variables más influyentes en las predicciones, ofreciendo insights valiosos para la toma de decisiones estratégicas.

Gráfico

El contenido generado por IA puede ser incorrecto.

Gráfico

El contenido generado por IA puede ser incorrecto.

# **Conclusiones**

Se desarrolló y evaluó modelos de ensamble que combinan los mejores clasificadores individuales obtenidos previamente para la predicción de morosidad en condominios residenciales. La implementación de técnicas de votación suave (Voting) y stacking heterogéneo permitió mejorar la robustez y precisión del sistema predictivo, alcanzando resultados superiores a los modelos individuales base.

* VotingClassifier mostró un desempeño sólido con un F1-score de 0.83, una precisión de 0.88 y un recall de 0.79 para la clase positiva (morosos). Su accuracy general fue del 88%, y el área bajo la curva ROC (AUC) alcanzó 0.95, demostrando una buena capacidad para discriminar entre morosos y no morosos.
* StackingClassifier superó ligeramente al Voting, con un F1-score de 0.85, precisiones de 1.00 en la clase negativa y 0.74 en la positiva, y un recall muy alto para la clase positiva (0.99). Esto indica que el stacking es especialmente efectivo para minimizar falsos negativos, crucial en la detección de morosidad. Su accuracy fue 87%, con un AUC de 0.95, reflejando excelente capacidad discriminativa.
* En la tabla comparativa, el modelo de stacking fue seleccionado como modelo final debido a su mejor F1-score, balance entre precisión y recall, y robustez general.

la curva ROC para el modelo de stacking confirma su alta capacidad para distinguir correctamente entre las clases, con una curva cercana a la esquina superior izquierda y un AUC de 0.95, lo que indica un desempeño muy sólido.

La curva Precisión-Recall muestra que el modelo mantiene alta precisión incluso cuando incrementa el recall, con un promedio de precisión (AP) de 0.88. Esto es crucial para mantener un equilibrio entre detectar correctamente los morosos y evitar falsos positivos.

La selección del modelo final con técnicas de ensamble, especialmente stacking, aporta una mejora significativa en la detección de morosidad, lo que es fundamental para la toma de decisiones oportunas en la administración de condominios residenciales. El modelo final es capaz de minimizar falsos negativos, reduciendo riesgos financieros y mejorando la eficiencia en la gestión de cobranza.

Además, las visualizaciones generadas (matrices de confusión, curvas ROC y precisión-recall, e importancia de características) facilitan la interpretación y validación del modelo, permitiendo a los equipos de negocio entender mejor el comportamiento predictivo y las variables más influyentes.

1. **Metodología de Evaluación de modelo RAG**

La evaluación de un sistema RAG se dividió en dos partes: la del retriever, que midió la calidad del contexto recuperado, y la del generador, que evaluó la calidad de las respuestas generadas a partir de ese contexto.

Cada componente se analizó con métricas específicas que permitieron identificar fortalezas y debilidades en la recuperación de información y en la generación de texto, asegurando así un diagnóstico más preciso del sistema completo.

### Evaluación del Retriever

La evaluación del retriever se enfocó en medir la calidad del contexto recuperado Para ello, se emplearon métricas de ranking que permitieron evaluar la efectividad del sistema al recuperar información relevante. Entre las más utilizadas se encontraron:

* Precisión (@k): Mide la proporción de documentos relevantes entre los k primeros documentos recuperados.
* Recall (@k): Indica la proporción de documentos relevantes recuperados respecto al total de documentos relevantes existentes.

Adicionalmente, para evaluar la precisión del retriever cuando se recuperaron múltiples documentos (top-k), se consideró la forma en que se realizó la comparación entre los documentos recuperados y los ground truth. Se aplicaó un esquema de comparación como:

* k a k: múltiples documentos recuperados vs. múltiples documentos esperados
* 3 documentos recuperados a 3 ground truths

Para este proyecto, orientado a la evaluación de un manual técnico, se optó por aplicar un esquema k a k, permitiendo realizar comparaciones más amplias mediante representaciones vectoriales o matriciales (n x n), según la cantidad de documentos recuperados y disponibles como ground truth.

También se utilizó la similitud del coseno como métrica base para comparar directamente los vectores del contexto recuperado con los del ground truth. Esta medida cuantificó el grado de similitud en el espacio vectorial y sirvió como fundamento para muchas de las métricas de ranking mencionadas, al ayudar a determinar si el contenido recuperado fue semánticamente relevante respecto al contexto esperado. De este modo, se obtuvo una evaluación más precisa del alineamiento entre la información recuperada y la información de referencia.

1. Evaluación del Generador

Esta parte puede no tener comparativa si la integración final es con google gemini llm, de ser así sólo se evluarían las diferentes combinacinoes de retriever integrada a 1 LLM

La evaluación del generador en un sistema RAG tuvo como objetivo medir la calidad de las respuestas generadas a partir del contexto recuperado. Esta etapa no solo buscó verificar la similitud entre la respuesta generada y la respuesta esperada (ground truth), sino también asegurar que la información generada fuera relevante, fiel al contexto y semánticamente coherente.

Los atributos principales que se evaluaron fueron:

* Relevancia: Qué tan pertinente fue la respuesta con respecto a la pregunta original.
* Fidelidad (faithfulness): Qué tan fiel fue la respuesta al contenido del contexto recuperado, es decir, si la respuesta reflejó correctamente la información disponible sin inventar datos (hallucinations).
* Fluidez: Qué tan natural y gramaticalmente correcta fue la respuesta generada.

Para evaluar estos atributos, se emplearon distintas métricas automáticas ampliamente utilizadas en tareas de generación de lenguaje natural:

* F1-score: Combinó precisión y recall a nivel de tokens, útil para evaluar la coincidencia de términos entre la respuesta generada y el ground truth, especialmente en respuestas más breves o categóricas.
* ROUGE-L: Midió la superposición de subsecuencias más largas comunes (longest common subsequence) entre la respuesta generada y la de referencia, útil para respuestas textuales más largas y estructuradas.
* BLEU: Comparó la coincidencia de n-gramas (normalmente hasta 4) entre la respuesta generada y una o más respuestas de referencia. Aunque originalmente diseñado para traducción automática, también se aplicó en generación de texto.
* BERTScore: Calculó la similitud semántica entre la respuesta generada y el ground truth utilizando embeddings obtenidos de un modelo BERT. A diferencia de métricas basadas en coincidencia exacta, permitió evaluar la proximidad semántica incluso si las frases no coincidían literalmente.

Muchas de estas métricas, en especial BERTScore, se apoyaron en representaciones vectoriales de las respuestas, y en algunos casos utilizaron similitud del coseno como base para determinar qué tan cercanos fueron los significados entre la respuesta generada y la esperada. Esto permitió capturar correspondencias más sutiles y semánticas, que van más allá de coincidencias superficiales de palabras.

### Definición y glosario de indicadores empleados

A white paper with black text

AI-generated content may be incorrect.

### Tabla comparativa de resultados:

La parte de resultados se obtuvo desde diferentes enfoques y con diferentes estrategicas enfocadas principalmente a evaluara el retriever. Las estrategias se evaluaron de la siguiente forma

### Evaluación exploratoria de 3 modelos diferentes utilizando el mismo splitter :

* all-MiniLM-L6-v2
* paraphrase-multilingual-mpnet-base-v2
* distiluse-base-multilingual-cased-v2

Para ester primer análisis exploratorio se estudiaron las diferentes preguntas, y se generaron 20 variaciones para cada pregunto, analizando inicialmente un universo de 240 preguntas posibles y en donde se evaluaron métricas como similitud de coseno y F1.

En total para cada modelo se estudiaron 2160 pares respuesta – contexto:

* 12 preguntas \* variaciones de pregunta \* 3 documentos extraídos \* 3 ground truths

A white rectangular sign with black text

AI-generated content may be incorrect.

A diagram with colorful rectangular objects

AI-generated content may be incorrect.

A graph with colorful rectangular shapes

AI-generated content may be incorrect.

A chart with colorful rectangular objects

AI-generated content may be incorrect.

NOTA: La relación de contenido releevante se generó estableciendo un thresold de .6, es decir si la simlitud entre el contexto y la respuesta era de .6 o más la respuesta obtenida por el retriever se consideró relevante. De este análisis prelimiar se puede observar que el modelo paraphrase-multilingual- mpnet-base-v2 presenta mejor desmpeño en los indicadores y distiluse-base-multilingual-cased-v2 no mostró resultados prometedores.

1. Evaluación de 3 modelos diferentes modificando el splitter (2 combinaciones posibles) e integrando el LLM de google gemini: En esta segunda se modificó la cantidad de preguntas a 14 y se analizaron las 14 respuestas para cada una de las 6 combinaciones modelo-splitter.

A screenshot of a table

AI-generated content may be incorrect.

De esta evaluación se puede observar que MiniLM- recursivo tiene los valores más altos en:

F1: mejor equilibrio entre precisión y recall

BLUE Y ROUGE-L: mejor cobertura léxica y textual

BERTSCORE: mejor alineamiento semántico con el ground truth

A graph with different colored lines

AI-generated content may be incorrect.

A graph with colorful lines and text

AI-generated content may be incorrect.

1. Evaluación del mejor modelo con diferentes estrategias de chunck size y chunk overlap (selección de hiperárametros)

Finalmente para la estrategia de evaluación de hiperpárametros es evaluaro diferentes combinaciones de chunk\_size y chunk overlaps para los 2 mejoes combinaciones:

* miniLM – recursivo
* mpnet -semántico

chunk\_sizes = [500, 750, 1000, 1250, 1500]

chunk\_overlaps = [0, 50, 100, 200]

De las diferentes combinaciones empleadas, la que mejor resultados obtuvo fue el modelo miniLM-recursivo

A black and white rectangular with black text

AI-generated content may be incorrect.

A screenshot of a table

AI-generated content may be incorrect.

Comparando vs los resultads previos, miniLM\_recursivo paso de .271 de F1 a .309 y Mpnet-semantico pasó de .275 a .277, por lo que miniLM semántico

# Fuentes:

Hernández-Sampieri, R., & Mendoza, C. (2023). Metodología de la investigación: Las rutas de la investigación cuantitativa, cualitativa y mixta. McGraw-Hill. <https://0-bc-vitalsource-com.biblioteca-ils.tec.mx/tenants/BIB_TECDEMTY/libraries?bookmeta_vbid=9786071520326>

Mukhiya, S., & Ahmed, U. (2020). Hands-On Exploratory Data Analysis with Python: Perform EDA Techniques to Understand, Summarize, and Investigate Your Data. Packt Publishing.

Visengeriyeva, L., Kammer, A., Bär, I., Kniesz, A., & Plöd, M. (2023). CRISP-ML(Q): The ML Lifecycle Process. INNOQ. <https://ml-ops.org/content/crisp-ml>

**MiCoto.** (2024). Manual de Usuario de la plataforma MiCoto. Documento interno (PDF).

**MiCoto.** (2024). Descripción funcional de MiCoto (dictado). Documento interno (PDF).

**MiCoto.** (2024). Base de datos de mensajes de usuarios (archivo CSV interno). Exportado desde la plataforma MiCoto para fines de análisis.

**MiCoto.mx.** (s.f.). *¿Quiénes somos?* Recuperado el 01/5/2025 de: <https://micoto.mx/#quienes-somos>