CARRERA DE ESPECIALIZACIÓN EN INTELIGENCIA ARTIFICIAL

MEMORIA DEL TRABAJO FINAL

Modelos de predicción de *overdue* y rotación para la redistribución del crédito financiero

Autor:
Lic. Paola Mariscal Zegarra

Director:
Ing. Yoel Yamil López (FIUBA)

Jurados:
Marcos Maillot (pertenencia)
Hanes Sciarrone (pertenencia)
Pablo Martín Gómez (pertenencia)

Este trabajo fue realizado en la Ciudad Autónoma de Buenos Aires, entre abril de 2022 y diciembre de 2022.

Resumen

La presente memoria describe el diseño e implementación de un algoritmo predictivo de mora y de rotación del crédito desarrollado para una importante compañía de consumo masivo que otorga créditos a sus clientes para la venta de sus productos. El sistema permite distinguir clientes que por su comportamiento podrían entrar en mora o aquellos que usen en mayor proporción el crédito asignado, lo que permitirá a la compañía mejorar la distribucion del presupuesto destinado a créditos.

Para poder abordar este trabajo se aplicaron conceptos de estadística, bases de datos y aprendizaje automático que permitieron llevar a cabo la implementación de un modelo estadístico basado en aprendizaje de máquina desde la extracción de datos hasta la generación de dos *scores*: riesgo y rotación.

Agradecimientos

A mi familia por apoyarme en cada nuevo desafío.

A mi director, Ing. Yoel Yamil López por su acompañamiento, orientación y por los conocimientos compartidos a lo largo de la carrera.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción general

En este capítulo se realiza un acercamiento a los conceptos básicos del aprendizaje automático y financieros. Asimismo, se menciona el estado del arte en un buró de crédito argentino, y por último se explica la motivación, alcance y objetivos del presente trabajo.

1.1. Inteligencia artificial en el ámbito financiero

El sector financiero fué y sigue siendo uno de los pioneros a la hora de implementar inteligencia artificial, una de las tecnologías más revolucionarias de los últimos tiempos.

La inteligencia artificial se refiere a sistemas o máquinas que imitan la inteligencia humana para realizar tareas, mejorando iterativamente a partir de la recopilación de información. Su objetivo es mejorar significativamente las capacidades humanas.

Actualmente la inteligencia artificial mejora el rendimiento y la productividad de las empresas mediante la automatización de los procesos y tareas que antes requerían esfuerzo humano o expertos dedicados a ello. Hoy en día, muchas aplicaciones de inteligencia artificial son utilizadas tanto por instituciones tradicionales como por *fintech* [1].

El *score* o *scoring* es un método estadístico que predice la probabilidad de *default* o incumplimiento de pago. Se representa con un número que generalmente oscila entre 1 y 999 es el principal factor que analizan los prestamistas ante solicitudes de crédito. Un *score* crediticio que utiliza inteligencia artificial es mucho más completo y sofisticado comparado con criterios tradicionales de *score* de crédito.

En la figura 1.1[1] se observa que la relación entre el *score* (representado por 'z') y el riesgo no es lineal, por lo que el cambio en el riesgo derivado de un cambio en el *score* depende de los valores que este último tome. Para valores del *score* muy bajos un aumento genera una rápida subida en la probabilidad de cumplimiento y una rápida disminución de la probabilidad de default, mientras que para valores de *score* altos, una mejora hace que la probabilidad de cumplimiento aumente poco y genera una leve caída en el riesgo.

[1] Imagen tomada de https://www.bcra.gob.ar/Pdfs/Publicaciones/CreditScoring.pdf

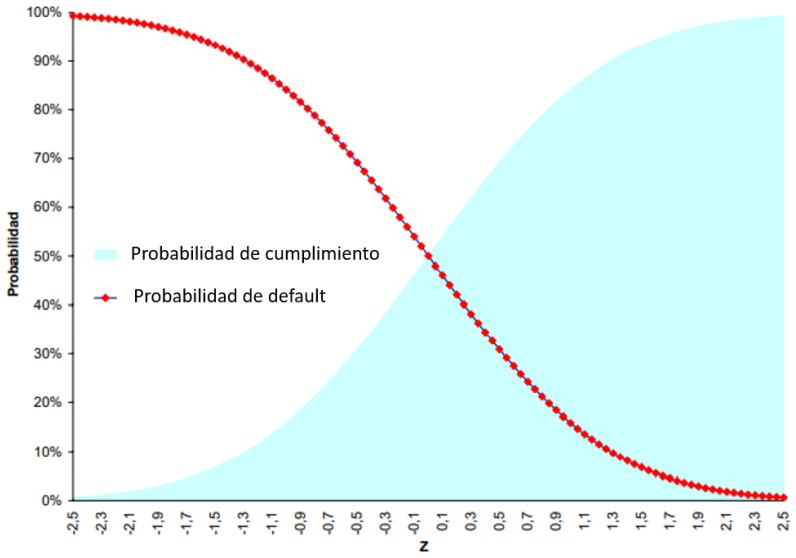

FIGURA 1.1. Relación entre el *score* y el riesgo.

Los bancos digitales y *fintech* utilizan algoritmos de aprendizaje automático o de máquina cuya importancia radica en la objetividad y la eliminación de sesgos para la toma de decisiones. Asimismo, la capacidad de cómputo de este tipo de sistemas, permite manejar grandes cantidades de datos en poco tiempo y la computación cognitiva ayuda a administrar datos estructurados y no estructurados. Los algoritmos analizan historiales de transacciones e identifican a tiempo signos de futuros problemas.

1.2. Estado del arte: caso Nosis

Nosis es una empresa fundada en la década del 80 con el objeto de brindar información de antecedentes comerciales, mercados financieros en línea y comercio exterior para aportar herramientas analíticas que faciliten la toma de decisiones. Como buró, cuenta con bases de datos exclusivas, información compartida por más de 100 entidades, información pública tanto del BCRA, ANSES, AFIP, entre otros e innovadoras técnicas analíticas con actualización constante de datos.

Una de sus principales unidades de negocio trabaja sobre informes comerciales que tienen como variable principal un *score* desarrollado por la misma empresa. El *score* de Nosis es un ejemplo de un *score* de riesgo. Brinda información sobre la probabilidad de *default* o mora del cliente consultado desde el momento de la consulta propiamente dicha y 12 meses hacia adelante. Detrás de este número único o calificación, hay más de 70 variables de información dentro del algoritmo desarrollado por la empresa que puso en consideración el endeudamiento histórico de varias fuentes de información y más de 600 atributos de datos. El algoritmo se procesa en el momento de la consulta e indica que cuanto más alto sea el *score*, existe una menor probabilidad de *default* o mora. Las bases de datos con las que cuenta Nosis cubren el 99.99 % del endeudamiento de una persona o empresa con el sistema financiero total a nivel país.

El presente trabajo se centró en el desarrollo de un *score* interno con variables e

información de los sistemas transaccionales de la compañía que por un lado permita automatizar el proceso manual llevado a cabo por el área de finanzas, y por el otro, permita evolucionar hacia una estrategia de política crediticia centrada en el comportamiento del cliente con la compañía.

En la figura 1.2 se observa el proceso previo al desarrollo del presente trabajo.

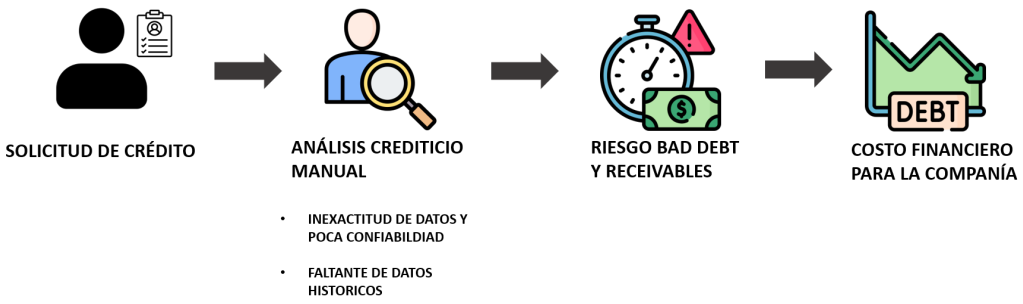

FIGURA 1.2. Proceso previo a la automatización.

1.3. Motivación

La principal motivación para la realización de este trabajo fue la eliminación de subjetividades en procesos de asignación de límite crediticio, la reducción de la deuda generada por los clientes morosos y la optimización del capital empleado (*receivables*) para la generación de mejores resultados, otorgando un mejor servicio de crédito a los clientes, junto a la adopción de tecnologías relacionadas al análisis predictivo en una empresa tradicional de muchos años y con procesos establecidos adversos al cambio.

La no existencia de una herramienta que permita objetivamente asignar límite de crédito a cada cliente significaba un alto riesgo asociado a la mora o incobrabilidad. De hecho, durante el año 2021 el *bad debt* o deuda incobrable ascendía casi al 30 % de la cartera de clientes con crédito. Por otro lado, la compañía no estaba cobrando interés alguno por los créditos cedidos y el P&L de crédito (*profit and loss*) arrojaba resultados negativos por un porcentaje considerable de clientes en mora sin un ROIC (*return on invested capital*)asociado.

En referencia a la opinión de los clientes en temáticas de crédito, se obtuvo un bajo NPS (*net promoter score*) en las encuestas de satisfacción de servicio enviadas. Sumado a lo anteriormente mencionado se encontraban a menudo errores manuales en análisis o asignación de crédito por causa de la no automatización de procesos. Existía la dependencia de personal específico solo para realizar un análisis de riesgo de cada cliente a partir de reglas de negocio definidas.

El presente trabajo fue el primer algoritmo desarrollado por el área de *data analytics* y el principal caso de uso dentro de la compañía, rompiendo así la brecha entre las áreas de negocio y de TI (tecnología de la información).

1.4. Objetivos y alcance

1.4.1. Objetivos

El propósito de este trabajo consistió esencialmente en el diseño, pruebas e implementación de una solución de inteligencia artificial que permite estimar la mora y la rotación del crédito de los clientes activos de la compañía, asegurando a una mejor distribución de los recursos crediticios, considerando distintos aspectos del cliente en cuanto a compras, pagos atrasados, deuda total vigente, cheques rechazados, antigüedad e historial de pagos, entre otras variables.

En la figura 1.3 se observa un diagrama general de la solución y las etapas que la componen.

FIGURA 1.3. Diagrama en bloques del sistema.

En el capítulo 3 se detalla el diseño y la implementación de las etapas que componen este sistema.

1.4.2. Alcance

El alcance de este trabajo estuvo orientado a desarrollar una solución de software que permite cubrir aspectos que giran en torno a los siguientes ejes:

1. Entendimiento del problema:
   - *Assesment* del proceso de asignación manual del límite de crédito.
   - Resultados esperados a partir de la implementación de la solución.
2. Aspectos relacionados al entendimiento del negocio:
   - Entendimiento de las restricciones asociadas al proceso.
   - Entendimiento de aspectos generales del negocio.
3. Aspectos relacionados a la adquisición y comprensión de datos:
   - *Assesment* de las fuentes de datos (modelos en *datawarehouse* y otras).
   - Exploración de los datos para determinar la calidad de la información.
4. Aspectos relacionados al modelado:
   - Diseño de características: generación de variables adicionales a partir de los datos sin procesar para facilitar el entrenamiento del modelo.

   - Entrenamiento del modelo: elección del modelo que responda a la pregunta de negocio con la máxima precisión, evaluando métricas de éxito.

5. Aspectos relacionados al despliegue:
   - Implementación del/los modelo/s en un entorno de producción o similar para el consumo por el usuario final o aplicaciones.

El alcance de este trabajo no cubre:

- El mantenimiento de la base de datos en la cual se aloja la información generadora de la entrada para el modelo de inteligencia artificial.
- La instalación y mantenimiento del *hardware* necesario para el procesamiento de información del modelo desarrollado.
- La disponibilización de la salida del algoritmo directamente en alguna aplicación de la compañía.

Capítulo 2

Introducción específica

En este capítulo se detallan los requerimientos asociados al desarrollo y la implementación de los modelos, así como el contexto y las necesidades de negocio. Finalmente se especificarán las tecnologías aplicadas en el desarrollo del trabajo, destacando los componentes de inteligencia artificial seleccionados para la predicción de la mora y la rotación.

2.1. Requerimientos asociados al desarrollo e implementación de/los modelo/s

2.1.1. Requerimientos asociados al desarrollo

1. Los códigos deben desarrollarse con herramientas de Microsoft Azure que es el servicio de computación en la nube utilizado por el área de data *analytics*.
2. El equipo de desarrollo tiene la potestad de utilizar el lenguaje de su conocimiento para el desarrollo del código del modelo.
3. Se utiliza GIT como repositorio para el control de versionado de código.

2.1.2. Requerimientos asociados a la implementación

1. Se utiliza Azure Databricks como herramienta tanto para el entrenamiento de los modelos como para su implementación.
2. Debe existir un archivo de ejecución principal llamado *main.py* y un *job* de ejecución automática de ese archivo en Databricks para la implementación *batch*, es decir sin supervisión.
3. Utilización de MLflow, una plataforma de código abierto para la administración del ciclo de vida de los modelos de aprendizaje automático, para el despliegue en producción y el seguimiento de los desarrollos dentro de la compañía.

2.2. Modelos de inteligencia artificial utilizados

Como se desarrollará en los próximos capitulos, el modelo elegido para el planteo de la solución fue la regresión logística. La regresión logística es un modelo de clasificación, también conocida como regresión logit o clasificador de máxima

entropía y se encuadra dentro de los modelos lineales generalizados. La regresión logística describe la probabilidad de que la variable objetivo pertenezca a una clase o a otra: dado un cierto valor límite, si el valor de la variable objetivo iguala o excede dicho valor, se devuelve como predicción la clase "positiva", caso contrario se devuelve como predicción la clase "negativa". Para ello se utiliza la función logística o también llamada función sigmoide, que siempre devuelve un valor entre 0 y 1. A continuación se detalla la función:

$$\text{función sigmoide} = f(x) = \frac{1}{1+e^{-x}}$$

En la figura 2.1 se puede observar la curva logística:

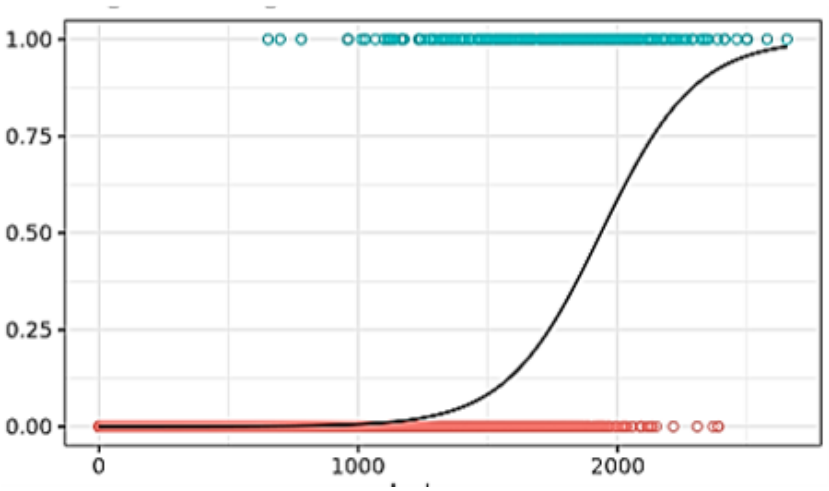

FIGURA 2.1. Gráfico función logística

La variable ”Y” a predecir es cualitativa debido a que el objetivo es predecir la probabilidad que un sujeto tome una determinada decisión de índole discreta, condicionada a ciertas variables explicativas y permitiendo identificar las características del sujeto con variables independientes cualitativas.

Para valores de x muy grandes positivos, el valor de e^{-x} es aproximadamente 0 por lo que el valor de la función sigmoide es 1. Para valores de x muy grandes negativos, el valor de e^{-x} tiende a infinito por lo que el valor de la función sigmoide es 0.

Un punto importante a tener en cuenta a la hora de implementar este modelo es que el set de datos con los que se trabaje debe ser normalizado porque de lo contrario podrían encontrarse indeterminaciones matemáticas.

2.3. Herramientas de software utilizadas

2.3.1. Espacio de trabajo: Databricks

Databricks es una plataforma que facilita la creación y ejecución de *pipelines* o canales de datos permitiendo la colaboración en proyectos de ciencia de datos y analíticos para la construcción e implementación de modelos de aprendizaje de máquina. Al mismo tiempo, ofrece soluciones de ingeniería de datos, administración, seguridad, gobernanza, entre otros. Está basado en código y estándares abiertos para maximizar su flexibilidad. El enfoque unificado simplifica la arquitectura de datos al eliminar los silos de datos que tradicionalmente separan la analítica, la inteligencia de negocios, la ciencia de datos y el aprendizaje automático. En la figura 2.2 se puede visualizar la arquitectura de Databricks:

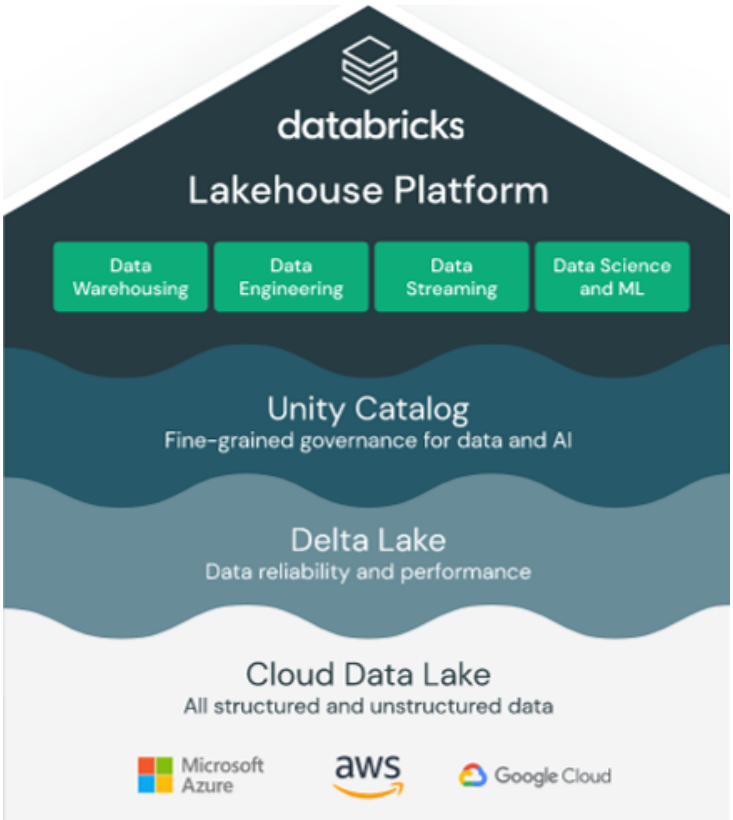

FIGURA 2.2. Arquitectura general de la plataforma Databricks.

En cuanto a la terminología utilizada dentro de la plataforma, se pueden detallar los siguientes conceptos clave:

- Espacio de trabajo: permiten organizar todo el trabajo realizado dentro de la plataforma similar a una estructura de carpetas que permite guardar archivos, bibliotecas utilizadas para la operación y manipulación de datos e incluso compartirlas con otros usuarios.

- *Notebooks*: se podría definir como conjunto de celdas que permiten ejecutar determinados comandos. Soporta cualquier tipo de lenguaje de programación como Scala, Python, R, SQL, entre otros. Los *notebook* deben estar conectados a un cluster para ejecutar comandos. Se pueden compartir o descargar de forma local.
- Librerías: son paquetes o módulos que proporcionan una funcionalidad adicional para la resolución de problemas de negocio. Facilitan la programación al proporcionar funcionalidades comunes, que han sido resueltas previamente por otros programadores.
- Tablas: son datos estructurados utilizados para el análisis, representados por una colección de filas y columnas almacenados como archivos de datos en el almacenamiento de objetos o base de datos.
- *Clusters*: son grupos de ordenadores que se tratan como un único ordenador. Permiten ejecutar *notebooks* o bibliotecas sobre un conjunto de datos. Cada clúster tiene controles de acceso para garantizar la seguridad de la información
- *Jobs*: Los *jobs* son la herramienta mediante la cual se puede programar la ejecución para que se produzca en un clúster ya existente o en un clúster propio.
- Aplicaciones: son integraciones de terceros con la plataforma Databricks.Entre ellas se encuentran aplicaciones como Power BI

2.3.2. Motor de análisis y procesamiento: Spark

Es un motor de análisis unificado para el procesamiento de datos a gran escala. Proporciona una interfaz de programación de aplicaciones de alto nivel, conocida también por la sigla API que es un conjunto de subrutinas, funciones y procedimientos que ofrece la biblioteca para ser utilizada por otro software como capa de abstracción. Es compatible con Spark SQL para consultas y procesamiento de datos en SQL, MLlib para el aprendizaje automático o GraphX para procesamiento de gráficos. Como se menciona en la sección 2.2, para el presente trabajo se utilizó la regresión logística como método para predecir la mora y la rotación de un cliente. A continuación se muestra mediante un ejemplo, cómo cargar un conjunto de datos de muestra, construir un modelo de regresión logística y hacer predicciones con el modelo resultante para calcular el error de entrenamiento:

```
1  from pyspark.mllib.classification import LogisticRegressionWithLBFGS,
       LogisticRegressionModel
2  from pyspark.mllib.regression import LabeledPoint
3
4  # Load and parse the data
5  def parsePoint(line):
6      values = [float(x) for x in line.split(' ')]
7      return LabeledPoint(values[0], values[1:])
8
9  data = sc.textFile("data/mllib/sample_svm_data.txt")
10 parsedData = data.map(parsePoint)
11
12 # Build the model
13 model = LogisticRegressionWithLBFGS.train(parsedData)
14
15 # Evaluating the model on training data
```

```
16 labelsAndPreds = parsedData.map(lambda p: (p.label, model.predict(p.
       features)))
17 trainErr = labelsAndPreds.filter(lambda lp: lp[0] != lp[1]).count() /
       float(parsedData.count())
18 print("Training Error = " + str(trainErr))
19
20 # Save and load model
21 model.save(sc, "target/tmp/pythonLogisticRegression")
22 sameModel = LogisticRegressionModel.load(sc,
23                                          "target/tmp
                                              pythonLogisticRegression")
```

2.3.3. Lenguaje de programación: Python

Python es un lenguaje de programación que permite trabajar rápidamente e integrar sistemas de forma más eficaz. Entre sus ventajas de uso se puede enumerar:

1. Amigable y fácil de aprender: existe una comunidad que organiza conferencias y reuniones para facilitar la colaboración entre los programadores. Cuenta con una extensa y detallada documentación que incluye especificaciones de versiones, tutoriales, librerías, módulos, reporte de errores, entre otros.
2. Variedad de aplicaciones: el índice de paquetes de *Python* (PyPI) alberga miles de módulos de terceros para *Python*. Tanto la biblioteca estándar de *Python* como los módulos aportados por la comunidad permiten un sinfín de posibilidades.
3. Código abierto: *Python* está desarrollado bajo una licencia de código abierto aprobada por la OSI (*Open Source Initiative*), lo que hace que se pueda utilizar y distribuir libremente, incluso para uso comercial. La licencia de *Python* es administrada por la *Python Software Foundation*.

2.3.4. Gestión del código y versiones: GitHub

GitHub es una plataforma de alojamiento de código para el control de versiones y la colaboración.Permite trabajar de forma colaborativa en proyectos desde cualquier lugar. A continuación, se enumeran los conceptos escenciales referidos a la plataforma:

- Repositorios: un repositorio contiene todos los archivos de un proyecto y el historial de revisiones de cada archivo. Existe la posibilidad de discutir y gestionar el trabajo del proyecto dentro del repositorio.
- Ramas: una rama es una versión paralela de un repositorio. Está contenida dentro del repositorio, pero no afecta a la rama primaria o principal, lo que le permite trabajar libremente sin interrumpir la versión "viva". Cuando se hayan realizado cambios, se puede fusionar dicha rama de nuevo con la rama principal para publicar los cambios o dicho de otra manera, realizar un *pull request*.
- *Commits*: un commit, o revisión", es un cambio individual en un archivo (o conjunto de archivos). Cuando se hace un commit para guardar el trabajo realizado, Git crea un ID único (también conocido como "SHA.º "*hash*") que permite mantener un registro de los cambios específicos confirmados junto

con quién los hizo y cuándo. Los *commits* suelen contener un mensaje de *commit* que es una breve descripción de los cambios realizados.

- Pull requests: las solicitudes de incorporación de cambios permiten comunicar acerca de los cambios insertados en una rama de un repositorio en GitHub. Una vez que se abre una solicitud de incorporación de cambios, se puede debatir y revisar los posibles cambios con los colaboradores y agregar confirmaciones de seguimiento antes de que los cambios se fusionen en la rama base.

2.3.5. Creación del proyecto, seguimiento y registro del modelo: MLFlow

MLFlow es una plataforma de código abierto para el ciclo de vida del aprendizaje automático incluyendo la experimentación, la reproducibilidad, el despliegue y un registro central de modelos. Trabaja perfectamente integrado con Python, Apache Spark, Azure Machine Learning, Databricks, entre otros. Aborda cuatro funciones principales:

1. MLflow Tracking: seguimiento de experimentos para registrar y comparar parámetros y resultados.
2. MLflow Projects: empaquetado del código de forma reutilizable y reproducible para compartirlo con otros científicos de datos o transferirlo a producción.
3. MLflow Models: gestión y despliegue de modelos desde una variedad de bibliotecas de aprendizaje automático a una variedad de plataformas de inferencia y servicio de modelos.
4. MLflow Model Registry: proporciona un almacén central de modelos para gestionar de forma colaborativa el ciclo de vida completo de un modelo MLflow, incluyendo el versionado de modelos, las transiciones de etapas y las anotaciones. MLflow es agnóstico a las librerías. Se puede utilizar con cualquier biblioteca de aprendizaje automático, y en cualquier lenguaje de programación, ya que todas las funciones son accesibles a través de una API REST y CLI.

2.3.6. Programación de flujos: Apache Airflow

Airflow es una plataforma creada por la comunidad para crear, programar y supervisar flujos de trabajo de forma programada. Se basa en las siguientes características:

- *Python* puro: con conocimientos básicos de Python, es posible es posible crear flujos de trabajo propios, incluyendo los formatos de fecha y hora para la programación y los bucles para generar dinámicamente las tareas. Esto permite mantener una flexibilidad total al construir flujos de trabajo.
- Interfaz de usuario útil: es posible gestionar, supervisar y programar los flujos de trabajo a través de una aplicación web sólida y moderna, contando con una visión completa del estado y los registros de las tareas completadas y en curso.

- Integraciones robustas: proporciona muchos operadores *plug-and-play* que están listos para ejecutar sus tareas en Google Cloud Platform, Amazon Web Services, Microsoft Azure y muchos otros servicios de terceros. Esto hace que *Airflow* sea fácil de aplicar a la infraestructura actual y de ampliar a las tecnologías de próxima generación.
- Fácil de usar: cualquier persona con conocimientos de Python puede desplegar un flujo de trabajo. *Apache Airflow* no limita el alcance de *pipelines*; se puede utilizar para construir modelos de ML, transferir datos, gestionar infraestructura, entre otros.
- Código abierto: donde se desee compartir una mejora o cambio, se puede hacer abriendo un *pull request*. Sencillo, sin barreras ni procedimientos prolongados.

Capítulo 3

Diseño e implementación

3.1. Arquitectura del sistema completo

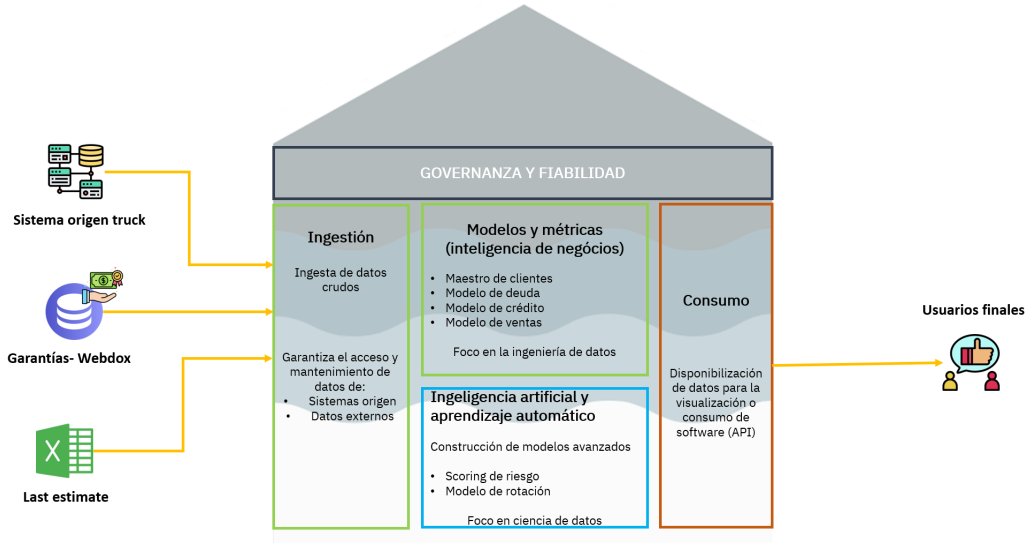

FIGURA 3.1. Arquitectura completa.

3.2. Generación y tratamiento de los datos

3.3. Automatización e implementación

Capítulo 4

Ensayos y resultados

4.1. Descripción del banco de pruebas

4.2. Modelos probados

4.3. Descripción de la salida

4.4. Caso de uso

Capítulo 5

Conclusiones

5.1. Conclusiones generales

5.2. Trabajo futuro

Bibliografía

[1] David Igual Molina. *Fintech: Lo que la tecnología hace por las finanzas*. Profit Editorial, 2016.