Grupo 3. Avances del proyecto(Entrega2)

ANA MARIA SOTO OROZCO, FELIPE DIEGO LOBATO DA SILVA, NICOLAS ALEJANDRO YEPES JOVEN y YOSELIN NIETO GIL

1. RESUMEN SOBRE EL PROBLEMA

1.1 Contexto y Definición del Problema:

El proyecto aborda un desafío común en la industria de productos de consumo masivo: la necesidad de anticipar los precios de venta de los productos durante las distintas temporadas del año (otoño, invierno, primavera y verano). Estas variaciones estacionales afectan los hábitos de compra y el precio de los productos, lo que puede complicar la planificación estratégica de las empresas. Si no se prevén adecuadamente, estos cambios pueden provocar problemas de inventario, como exceso o falta de productos, pérdidas de ventas y costos adicionales. El objetivo de este proyecto es desarrollar un modelo que, utilizando datos históricos de ventas, estime un precio de venta óptimo por temporada, ayudando a las empresas a mejorar su planificación estratégica y eficiencia operativa.

1.2 Pregunta de Negocio y Alcance del Proyecto:

La pregunta central del proyecto es: ¿Cómo predecir el precio de venta estacional de productos de consumo masivo para optimizar la gestión de inventarios y mejorar la eficiencia operativa?

Este proyecto se limita a productos de consumo masivo vendidos a través de plataformas de e-commerce, en particular Olist Store, durante el año 2018. El modelo debe prever cuáles productos tendrán un mayor precio de venta en cada temporada, analizando patrones estacionales y de comportamiento del consumidor. La solución está dirigida a empresas que buscan ajustar sus inventarios y maximizar la disponibilidad de productos, considerando el precio estacional estimado y la demanda anticipada, para una mayor efectividad en sus estrategias comerciales.

1.3 Datos a Emplear y Preparación Inicial:

El proyecto empleará datos históricos de Olist Store, que incluyen información detallada sobre transacciones, productos, clientes, vendedores y reseñas. Las fuentes de datos principales se obtuvieron de Kaggle, complementadas con datos de códigos postales (CEP) adquiridos de un proveedor para enriquecer los análisis geoespaciales. La preparación de datos incluyó varios pasos:

a. **Estandarización y Transformación**: Se unificaron los formatos de los CEP y se convirtieron los archivos a Parquet para mejorar el rendimiento.
b. **Geolocalización**: Se integraron y ajustaron datos de geolocalización mediante un cruce de archivos y el uso de la API de Nominatim para obtener coordenadas precisas.
c. **Consolidación:** Los datos enriquecidos se consolidaron en un archivo optimizado para análisis espaciales, con las coordenadas ajustadas en formato de alta precisión.
d. Transformación final: Con los datos consolidados, se homologan los nombres de los campos al español para facilitar su interpretación. A continuación, utilizando el campo de fecha de compra del pedido, se filtra la base de datos para incluir únicamente el año más reciente disponible, en este caso, 2018. Posteriormente, se crea el campo temporada en función del mes de la fecha de compra, categorizando cada registro en una de las cuatro estaciones del año: otoño, invierno, verano o primavera. Esta transformación permitirá analizar y prever variaciones estacionales en los precios de venta de forma más precisa. Se obtiene una base con 63.215 registros y 33 columnas.

A continuación, se detalla la información de la base final:

Tabla 1" Tabla de metadatos de base final"

Nombre de la Columna	Descripción	Tipo de Dato	Ejemplo del Dato	Clave
id_pedido	Identificador único del pedido	string	"995392413cee616c1..."	Clave primaria
id_cliente	Identificador único del cliente	string	"4bf2490c4245cdb25a..."	Clave foránea (customers.customer_id)
estado_pedido	Estado del pedido (ej. entregado, cancelado)	string	"entregado"	
fecha_pedido	Fecha en que se realizó la compra	timestamp	"2017-09-04"	
hora_compra_pedido	hora en que se realizó la compra	string	"22:43:54"	
fecha_aprobacion_pedido	Fecha y hora en que se aprobó el pedido	timestamp	"2017-09-04 22:43:54"	
fecha_entrega_estimada	Fecha estimada de entrega del pedido	timestamp	"2017-09-27 00:00:00"	
id_item_pedido	Identificador del ítem dentro del pedido	integer	1	Clave primaria compuesta junto a id_pedido
id_producto	Identificador único del producto	string	"4lebbrb7a41c44632..."	Clave foránea (products.product_id)
id_vendedor	Identificador único del vendedor	string	"7a67c85e85bbc2e85..."	Clave foránea (sellers.seller_id)
fecha_limite_envio	Fecha límite para el envío	timestamp	"2017-05-22 16:05:44"	
precio	Precio del producto	doble	109.99	
valor_flete	Valor del envío	doble	18.02	

secuencia_pago	Secuencia del pago dentro del pedido	integer	1	Clave primaria compuesta junto a id_pedido
tipo_pago	Tipo de pago utilizado	string	"tarjeta_credito"	
cuotas_pago	Número de cuotas del pago	integer	8	
valor_pago	Valor del pago	doble	99.33	
id_unico_cliente	Identificador único y persistente del cliente	string	"15ee900ec703c9a10"	
codigo_postal_cliente	Prefijo del código postal del cliente	string	"68590"	Clave foránea (geo.cep_prefix)
ciudad_cliente	Ciudad del cliente	string	"jacunda"	
estado_cliente	Estado del cliente	string	"PA"	
nombre_categoria_producto	Categoría del producto	string	"perfumaria"	
longitud_nombre_producto	Longitud del nombre del producto	integer	40	
longitud_descripcion_producto	Longitud de la descripción del producto	integer	287	
cantidad_fotos_producto	Cantidad de fotos del producto	integer	1	
peso_producto_g	Peso del producto en gramos	integer	225	
largo_producto_cm	Longitud del producto en centímetros	integer	16	
altura_producto_cm	Altura del producto en centímetros	integer	10	
ancho_producto_cm	Ancho del producto en centímetros	integer	14	

codigo_postal_vendedor	Prefijo del código postal del vendedor	string	"13023"	Clave foránea (geo.cep_prefix)
ciudad_vendedor	Ciudad del vendedor	string	"campinas"	
estado_vendedor	Estado del vendedor	string	"SP"	
temporada	Estación del año de acuerdo al mes de compra del pedido (Otoño, Invierno, Verano, Primavera)	string	"Verano"	

2. MODELOS DESARROLLADOS Y SU EVALUACIÓN

Con la intensión de determinar el mejor modelo para resolver las necesidades de la empresa Olist Store, de definió desarrollar modelo que predice el precio de productos de acuerdo con la temporada en la que se compran, la ubicación del cliente, la categoría del producto y sus características físicas. Esta información resulta valiosa para la empresa como una herramienta que complementa la planeación contable, financiera y logística de la empresa, así como para la definición de estrategias de marketing ajustadas a los precios de venta de sus productos a lo largo del año.

Proceso de Entrenamiento:

- Cada modelo fue implementado dentro de un pipeline que integra preprocesamiento y entrenamiento, facilitando la escalabilidad y reproducibilidad del flujo de trabajo.
- Los datos se dividieron en conjuntos de entrenamiento y prueba utilizando una proporción de 80-20, con una semilla (random_state=42) para garantizar la consistencia en cada iteración.
- Se aplicó una búsqueda de hiperparámetros para cada modelo usando GridSearchCV o RandomizedSearchCV, asegurando así la selección óptima de parámetros para minimizar errores y mejorar la precisión.

Selección de Características:

- Las características utilizadas incluyen tanto atributos numéricos (como el peso, largo, altura y ancho del producto) como categóricos (temporada, categoría del producto, ciudad y estado del cliente, y el ID del producto).
- El preprocesamiento incluyó la estandarización de variables numéricas mediante StandardScaler y la codificación de variables categóricas con OneHotEncoder, facilitando una entrada balanceada y normalizada para los modelos.
- La selección de estas características se basó en su relevancia teórica para la predicción del precio del producto según factores como la estacionalidad y las propiedades físicas del producto.

Evaluación de Modelos:

- Los modelos fueron evaluados utilizando métricas de error y precisión, seleccionando la raíz del error cuadrático medio (RMSE), el error absoluto medio (MAE) y el coeficiente de determinación (R^2) para capturar diferentes aspectos del rendimiento.
- Se aplicó validación cruzada para obtener un RMSE promedio que refleje la robustez del modelo en diferentes particiones del conjunto de datos, disminuyendo así la posibilidad de overfitting o underfitting.
- Los modelos probados incluyen XGBRegressor, RandomForestRegressor, LinearRegression, ElasticNet y KNeighborsRegressor. En términos de rendimiento, el modelo de regresión lineal obtuvo el mejor RMSE (118.01) y el mayor R^2 (0.6336), lo cual indica que es el modelo más adecuado para este caso(Documentación de entren…)(modelos).

1. **Modelo con XGBRegressor:**
    - **Descripción**: Se implementa un pipeline que incluye preprocesamiento y el modelo XGBRegressor. Se realiza una búsqueda de hiperparámetros con GridSearchCV.
    - **Parámetros óptimos**: {'colsample_bytree': 0.8, 'gamma': 0.1, 'learning_rate': 0.1, 'max_depth': 9, 'n_estimators': 300, 'reg_alpha': 0.01, 'subsample': 0.8}
    - **Métricas**:
        - **RMSE**: 132.67
        - **MAE**: 54.75
        - **R^2**: 0.5368
        - **RMSE (validación cruzada)**: 127.56
2. **Modelo con RandomForestRegressor:**
    - **Descripción**: Similar al anterior, este pipeline utiliza RandomForestRegressor y RandomizedSearchCV para ajustar hiperparámetros.
    - **Parámetros óptimos**: {'n_estimators': 100, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_features': 'log2', 'max_depth': None, 'bootstrap': False}
    - **Métricas**:
        - **RMSE**: 135.53
        - **MAE**: 43.09
        - **R^2**: 0.5167
        - **RMSE (validación cruzada)**: 132.04
3. **Modelo con LinearRegression:**
    - **Descripción**: Modelo lineal básico, sin ajuste de hiperparámetros. Incluye preprocesamiento mediante un pipeline.
    - **Métricas**:
        - **RMSE**: 118.01
        - **MAE**: 25.55
        - **R^2**: 0.6336

        - **RMSE (validación cruzada)**: 114.60

4. **Modelo con ElasticNet:**
    - **Descripción**: Modelo de regresión penalizada (ElasticNet) que también se entrena en pipeline.
    - **Métricas**:
        - **RMSE**: 183.53
        - **MAE**: 81.10
        - **R^2**: 0.1137
        - **RMSE (validación cruzada)**: 171.00
5. **Modelo con KNeighborsRegressor:**
    - **Descripción**: Modelo KNeighborsRegressor con ajuste de hiperparámetros mediante GridSearchCV.
    - **Parámetros óptimos**: {'n_neighbors': 5, 'p': 1, 'weights': 'distance'}
    - **Métricas**:
        - **RMSE**: 142.54
        - **MAE**: 39.61
        - **R^2**: 0.4654
        - **RMSE (validación cruzada)**: 137.98

Modelo con Mejor Desempeño

Basado en el RMSE y el R^2, el modelo de **regresión lineal (LinearRegression) tiene el mejor desempeño** con un **RMSE de 118.01** y un **R^2 de 0.6336**. Este modelo proporciona las predicciones más precisas en este conjunto de pruebas y validación cruzada.

3. OBSERVACIONES Y CONCLUSIONES SOBRE LOS MODELOS

Para responder a la pregunta de negocio de predecir los precios estacionales de productos de consumo masivo, decidimos centrarnos en el modelo de regresión lineal, que había demostrado un buen rendimiento en la fase inicial descrita en el punto 2. Con este modelo, creamos un experimento en MLflow específicamente para monitorear y comparar varias configuraciones y ajustes, asegurando así que podríamos determinar la mejor versión para nuestro caso de uso. Este experimento fue fundamental, ya que nos permitió registrar y visualizar los resultados de cada variación en un entorno controlado y reproducible, proporcionando una comprensión clara de cómo cada ajuste afectaba el desempeño del modelo.

Dentro de este experimento, generamos tres versiones distintas del modelo, que correspondieron a tres códigos diferentes, cada uno enfocado en una configuración específica de preprocesamiento y ajuste de hiperparámetros para maximizar la precisión y la eficiencia. A continuación, detallamos cada uno de estos códigos y los resultados obtenidos:

Código 1: Este código representa la versión inicial y más básica del modelo de regresión lineal, que fue creada originalmente en la sección de modelos descrita anteriormente. Sin embargo, para este experimento, se adaptó para ejecutarse en el entorno de Databricks e integrarse con MLflow para un seguimiento sistemático. En esta configuración, utilizamos un pipeline simple que incluía la normalización de las variables numéricas (peso, dimensiones del producto) y

la codificación one-hot para variables categóricas (temporada, categoría del producto, entre otras). Este enfoque buscaba capturar las relaciones principales entre las variables sin añadir complejidad excesiva al modelo. Los resultados obtenidos para esta versión fueron prometedores, con un RMSE de 118.01, MAE de 25.55 y un R^2 de 0.63, además de una validación cruzada que mostró un RMSE promedio de 114.60, indicando robustez y consistencia. Este rendimiento destacado convirtió a esta versión en una fuerte candidata para ser la solución final.

Código 2: La segunda versión del modelo buscó explorar un preprocesamiento más sofisticado de las variables categóricas, con el fin de mejorar la captura de patrones latentes. Aquí, decidimos combinar dos técnicas de codificación: OneHotEncoder para algunas variables categóricas con menor número de categorías, y TargetEncoder para variables como la ciudad del cliente y el identificador del producto, que tienen un mayor número de categorías y podrían beneficiarse de una codificación basada en la media del objetivo (precio). Este enfoque introdujo mayor flexibilidad al manejar variables de alta cardinalidad, permitiendo que el modelo capturara mejor las sutilezas de cada categoría. Aunque esta configuración no trajo una mejora significativa en los resultados comparada con la primera versión, añadió un valor teórico y práctico importante, mostrando su utilidad especialmente en casos donde hay gran variabilidad en las categorías.

Código 3: La tercera versión fue un intento de optimizar el modelo mediante la selección de características. En esta versión, incorporamos el método SelectKBest con una función de puntuación basada en regresión (f_regression) para seleccionar las cinco variables más relevantes para la predicción del precio. La motivación detrás de este enfoque fue reducir el ruido de variables menos significativas, optimizando el modelo tanto en términos de rendimiento como de eficiencia computacional. Los resultados de esta versión fueron similares a los de las versiones anteriores en términos de RMSE y R^2, pero la reducción de complejidad computacional la convierte en una opción interesante para aplicaciones que requieren un tiempo de procesamiento más rápido.

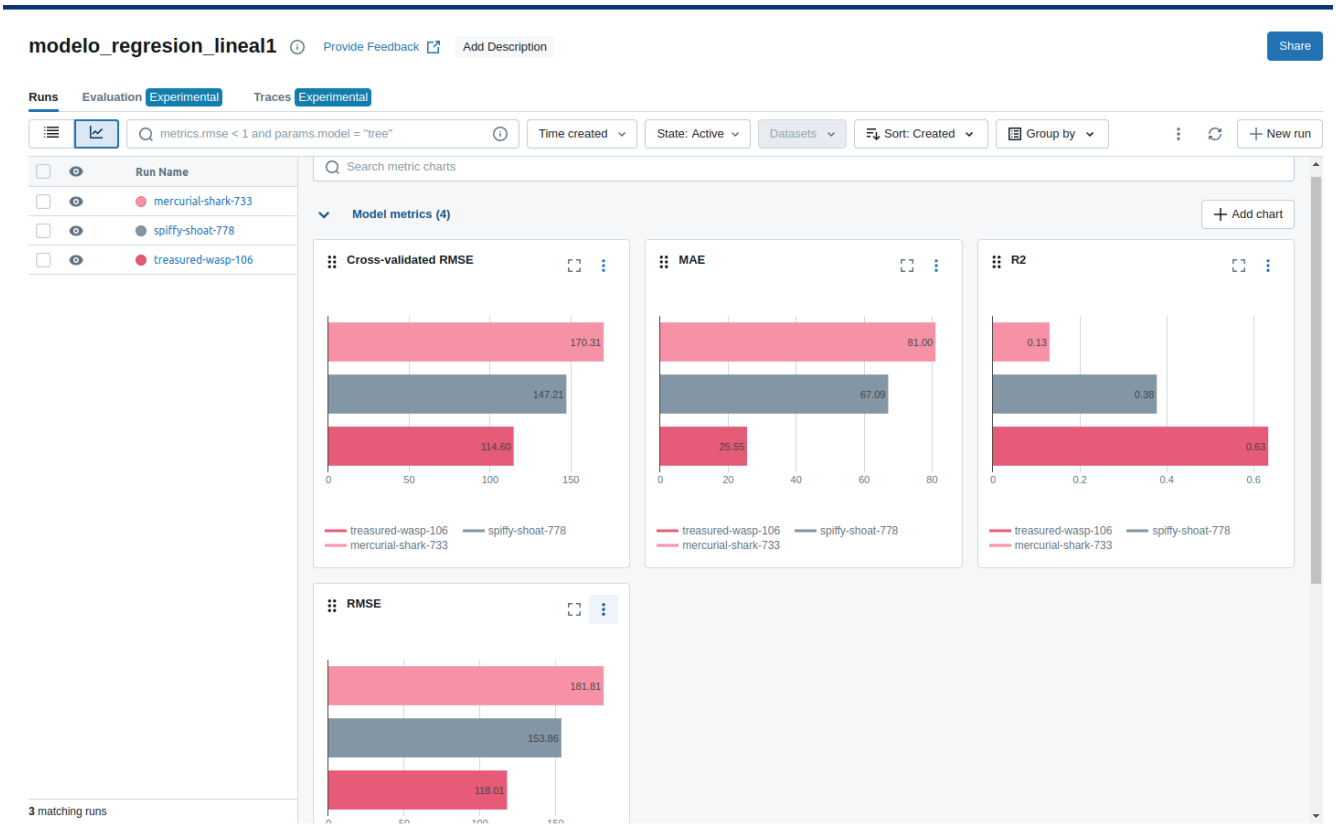


Figura 1 - Resultados del experimento de distintas configuraciones del modelo de regresión lineal en MLflow. La imagen muestra una comparación de rendimiento entre las tres versiones del modelo, destacando la configuración con mejor desempeño en términos de precisión y consistencia.

Comparación y Resultados Consolidados en Mlflow

La visualización de los resultados en MLflow nos permitió realizar un análisis comparativo claro entre las tres versiones del modelo. Observamos que, entre todas las versiones, el Código 1 mantuvo el mejor equilibrio entre precisión y simplicidad. Con un RMSE de 118.01, MAE de 25.55, y una validación cruzada con RMSE promedio de 114.60, destacó como el modelo más consistente y eficaz para el objetivo de predicción. El uso de MLflow no solo facilitó el registro detallado de las métricas, sino que también permitió un análisis visual y objetivo de los resultados, confirmando la robustez y eficiencia del Código 1 en comparación con las otras versiones.

Conclusión

Con base en los resultados de las tres versiones, concluimos que la primera configuración, representada por el Código 1, es la más adecuada para el proyecto. Este modelo de regresión lineal, ahora adaptado al entorno de Databricks y gestionado a través de MLflow, cumple con los requisitos de precisión y simplicidad, además de permitir un fácil seguimiento de versiones y métricas para futuras mejoras. La experiencia con MLflow fue fundamental para el éxito del experimento, permitiéndonos documentar y rastrear cada etapa del proceso de manera eficiente y clara. Así, el Código 1 se posiciona como la elección final para implementación, atendiendo a los objetivos de predicción de precios estacionales de forma precisa y práctica.

4. TABLERO DESARROLLADO

Para la elaboración del Dash inicial del proyecto, se creó un archivo jupyther notebook para el dash el cual se basa temporalmente en el archivo base_dash.csv, que contiene las características y predicciones realizadas por el modelo de regresión lineal sobre df_baseFinal.parquet, seleccionado para las predicciones del precio. El tablero muestra de acuerdo con la temporada seleccionada, la tabla con las categorías de producto y su demanda en esa temporada también muestra un gráfico de barras con el precio promedio por categoría según la temporada, otra parte del Dash sobre las predicciones que se conectará al modelo, busca que el cliente (vendedores) seleccione algunas características usadas como la features del modelo y se estime un precio aproximado.

Se creó una instancia en AWS EC2 y se configuró security group para permitir el acceso al dash. Se creó la carpeta “mi_proyecto dash” en la máquina virtual y se cargó la base de datos “base_dash.csv” y se creó el “Dash_despliegue_sem5.py” con la configuración del dash, se ejecutó y el dash quedó operativo.

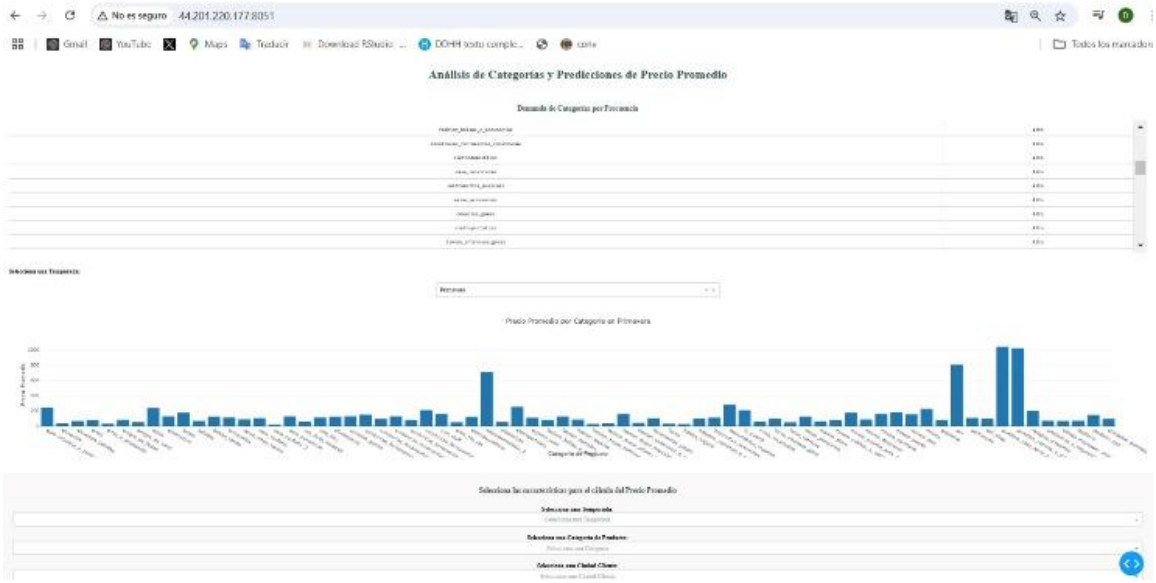


Figura 2 - Parte A del Dash. Análisis de categorías

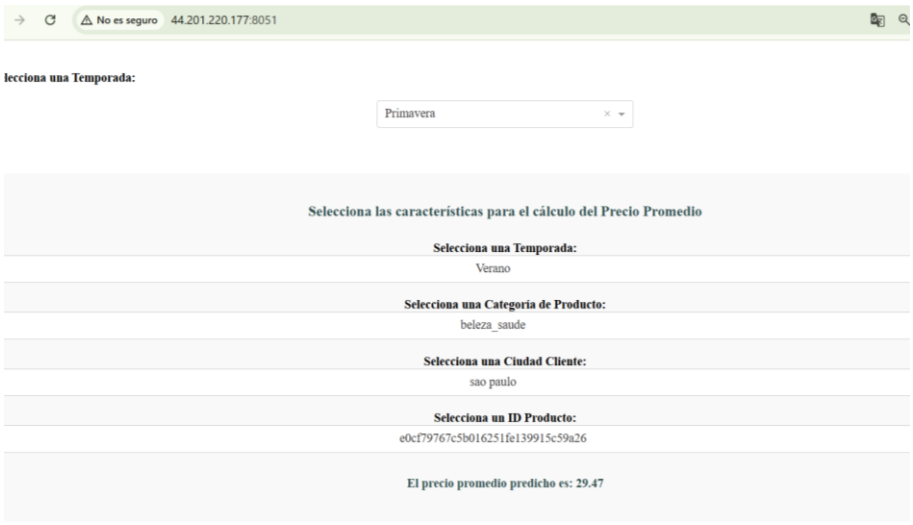


Figura 3. Parte B -Dash. Predicciones

5. Responsabilidades y Contribuciones de los Integrantes

Ana María Soto Orozco:
Asumió el rol de coordinadora del equipo, organizando las tareas y centralizando cada una de las entregas del proyecto. Su trabajo incluyó la realización del análisis exploratorio de los datos, lo que permitió comprender la estructura y características principales de la información. También fue responsable de la limpieza y procesamiento de los datos, asegurando que estuvieran en condiciones óptimas para su uso en el modelado. También realizó el cruce entre las diferentes bases de datos para generar una base final con la información requerida, que sirvió como insumo para las etapas siguientes. Adicionalmente, colaboró en las actividades de modelación junto con Nicolás.

Nicolás Alejandro Yepes Joven:
Fue el encargado de la fase de modelación del proyecto. Su trabajo incluyó la creación de modelos de aprendizaje automático, la evaluación de métricas de desempeño, y la comparación entre distintos enfoques para seleccionar el más adecuado. Realizó una exhaustiva búsqueda de hiperparámetros para optimizar el mejor modelo y llevó a cabo pruebas fuera de tiempo para asegurar la generalización del modelo seleccionado. Gracias a sus aportes, se logró definir una solución analítica robusta y alineada con los objetivos del proyecto.

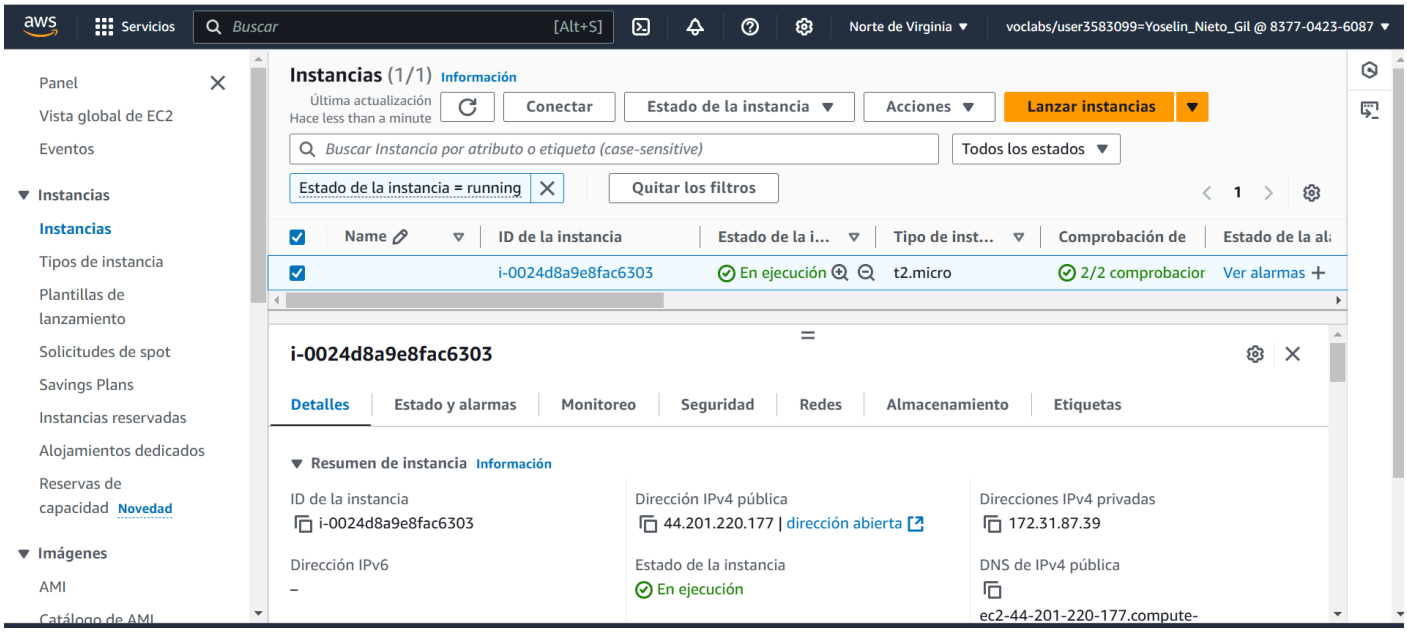
Felipe Diego Lobato da Silva:
Fue el responsable de establecer la infraestructura para el despliegue de los modelos. Implementó la estructura en MLflow para facilitar el seguimiento de los experimentos y creó la instancia en AWS necesaria para el despliegue del proyecto. Además, configuró un perfil en Databricks para comparar los experimentos de los modelos, permitiendo gestionar y evaluar los resultados de manera centralizada. Su trabajo en esta fase fue clave para asegurar la escalabilidad y trazabilidad del despliegue.

Yoselin Nieto Gil:
Se encargó de la fase de visualización, desarrollando el dashboard del proyecto. Se aseguró de que el tablero cumpliera con todos los requisitos planteados y de que presentara los resultados de forma clara y útil para los usuarios finales. Asimismo, fue la revisora final de la documentación, verificando la coherencia y completitud del reporte para garantizar que reflejara adecuadamente el trabajo del equipo y los hallazgos del proyecto.Coordinación y
Reuniones de Equipo

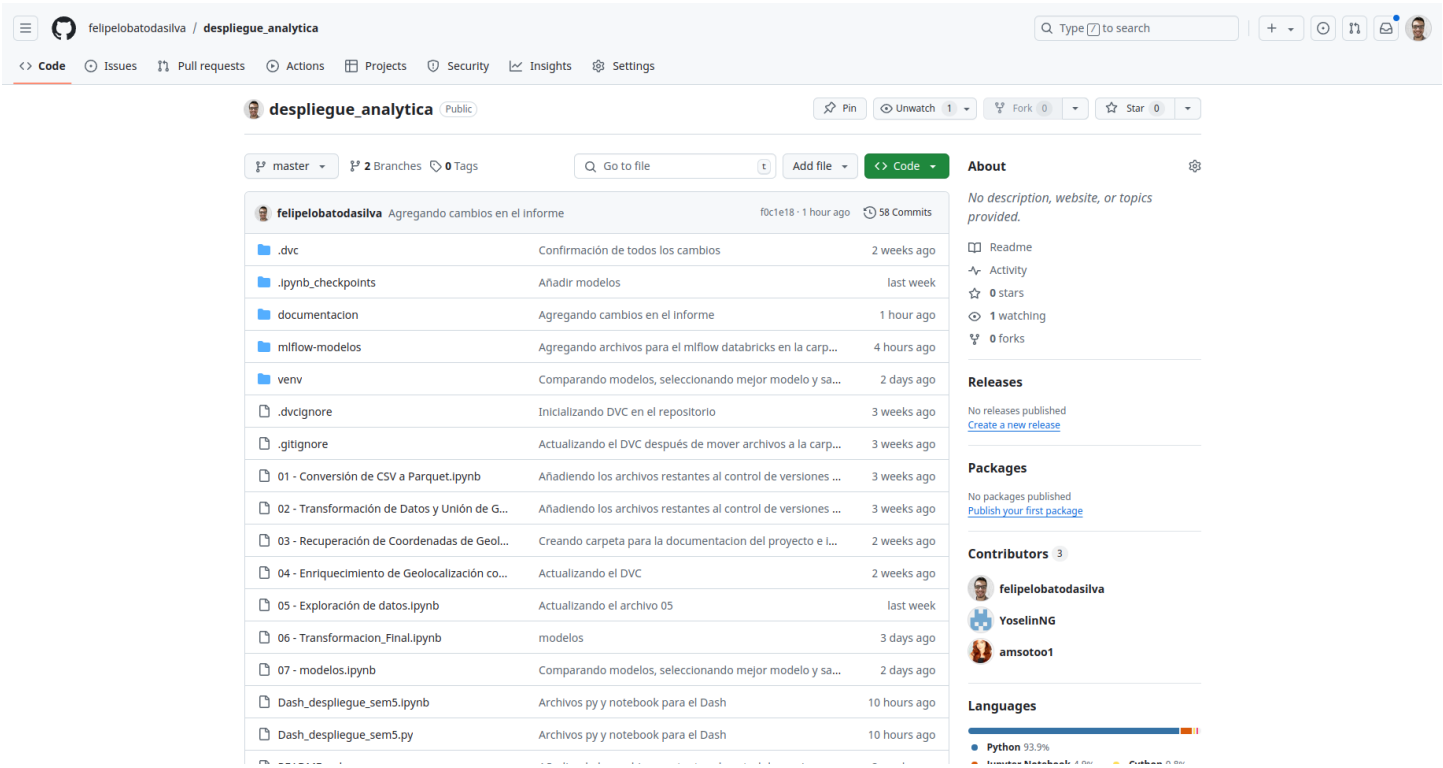
El equipo mantuvo reuniones periódicas para controlar el avance de cada tarea, analizar los resultados intermedios, y asegurar la cohesión en cada fase del proyecto. En estas sesiones, se compartieron avances, se resolvieron dudas, y se sacaron conclusiones sobre el desarrollo del trabajo. Todos los integrantes documentaron sus tareas para facilitar la trazabilidad del proyecto y enriquecer la documentación final.

6. APENDICE

Instancia en AWS EC2 para el Tablero



Pantalla Inicial del repositorio General



Link:

Aplicación mlflow – http://3.84.19.167:8050/

Ruta de los scripts del modelo: GitHub – felipelobatodasilva/despliegue_analytica/mlflow-modelos

Ruta del repo general - https://github.com/felipelobatodasilva/despliegue_analytica

Ruta del dash en linea - http://44.201.220.177:8051