Prueba Analítica: Modelo Opciones de Pago

Aspirante: Walter Arboleda Castañeda
Email: warboled@bancolombia.com.co

Objetivo:

Desarrollar una solución analítica E2E que permita poner en producción un modelo que genere un indicador de propensión a la aceptación de opción de pago de cada obligación de un cliente en mora.

Nota:

- Todos los scripts mencionados a continuación están alojados en el repositorio, son funcionales y desarrollados para esta prueba a excepción del CD.
- Los experimentos del modelo están alojados en tags del repositorio con lo cual permite navegar en el código ejecutado para cada uno de ellos

Desarrollo

Estrategia:

La estrategia consistió en la automatización de las diferentes fases de la solución, donde se utilizan herramientas y se construyen scripts propios para agilizar las actividades de: exploración, selección, limpieza y construcción de variables; del mismo modo para la ejecución de múltiples experimentos en la selección del modelo más adecuado, y algunas funciones adicionales para la fase de monitoreo y despliegue del modelo.

De esta forma se plantea una solución con componentes reutilizables y parametrizables.

Herramientas utilizadas:

- **PyCaret**: Es una biblioteca de machine learning en Python que simplifica la experimentación con modelos, automatizando tareas como preprocesamiento de datos, selección de modelos, ajuste de hiperparámetros y evaluación.
  Home - PyCaret

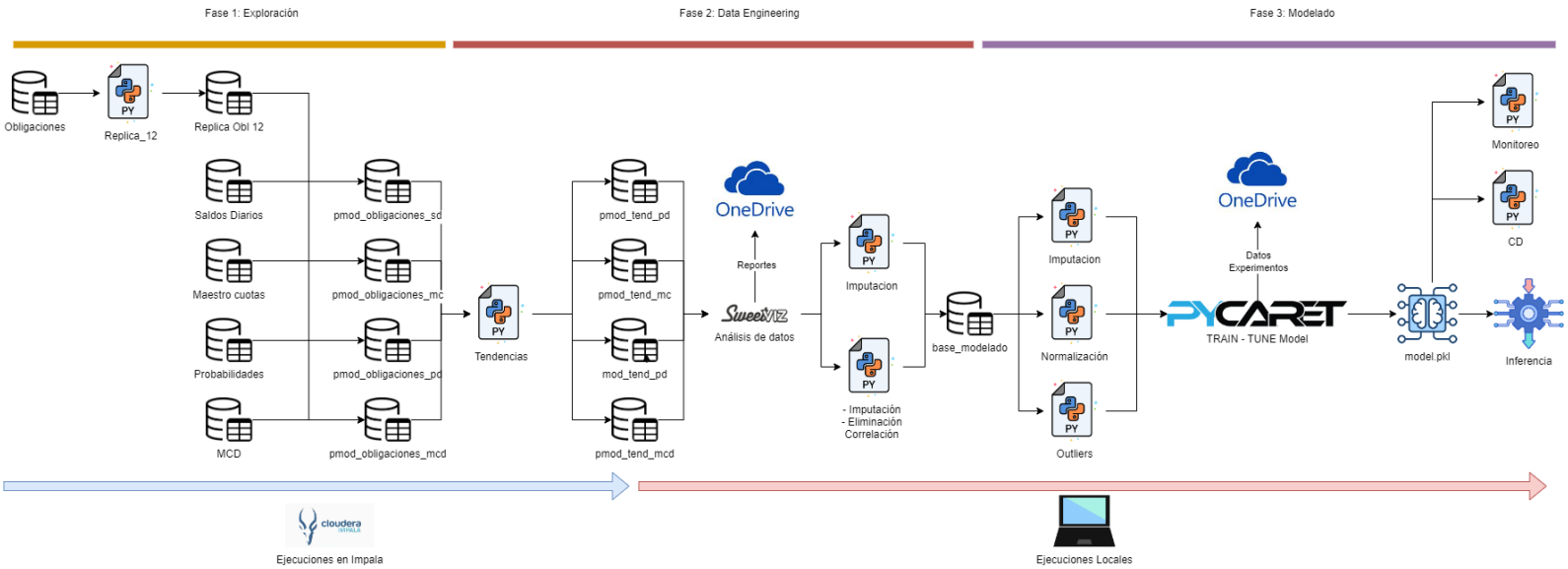
- **Sweetviz**: Es una herramienta en Python para la visualización exploratoria de datos, que genera automáticamente un reporte detallado y visualmente atractivo del conjunto de datos, facilitando el análisis inicial y la comprensión de las características de los datos. sweetviz · PyPI

Lenguajes:

Todo el desarrollo es realizado bajo los lenguajes de Python y SQL, haciendo provecho de las librerías que sincronizan ambos (impala-helper) y permiten fácil conexión con la fuente de datos.

Diagrama Solución:

A continuación, se presenta el flujo de la solución y las diferentes fases donde se utilizaron las herramientas, detallando donde se construyen los scripts en cada una de las fases.



Como se puede observar en el diagrama esta dividido por tres fases las cuales se detallan a continuación

Nota: Nótese que en la parte inferior indica que ejecuciones se utilizaron con recursos impala y cuales con recursos de maquina local

Fase 1: Exploración

Esta fase consistió en el análisis inicial de las fuentes proporcionadas donde se usaron las siguientes:

- prueba_op_base_pivot_var_rpta_alt_enmascarado_trtest: Obligaciones

- prueba_op_master_customer_data_enmascarado_completa: MCD
- prueba_op_probabilidad_oblig_base_hist_enmascarado_completa: Probabilidades
- prueba_op_maestra_cuotas_pagos_mes_hist_enmascarado_completa: Maestro Cuotas
- prueba_op_saldos_diarios_cob_enmascarado_completa: Saldos Diarios

Siendo el problema de propensión a seleccionar una opción de pago, lo primero es seleccionar las variables que informen sobre la evolución y el comportamiento de la obligación, las cuales son: cuotas, proporciones de pago, moras, marcas, ajustes. Esto dada la suposición que un cliente con mayor variación en su comportamiento de pago puede ser mas propenso a optar por un ajuste en su obligación.

Las variables seleccionadas de cada fuente son:

Fuente	Variable	Observación
Saldos D	sld_cap_final	Al ser una tabla con registro diario se convierten los valores a datos mensuales sacando de cada periodo los valores AVG, MAX y MIN
	nueva_altura_mora	
	vlr_obligacion	
	vlr_vencido	
Maestro Cuotas	valor_cuota_mes	Las variables marca_pago y ajustes banco son 2 de las 3 variables categóricas utilizadas
	pago_total	
	porc_pago	
	marca_pago	
	ajustes_banco	
MCD	total_ing	Se utiliza esta información con el objetivo de ver la influencia del flujo de dinero del cliente en la variable respuesta
	tot_activos	
	egresos_mes	
	tot_patrimonio	
Probabilidades	prob_propension	La variable lote es una de las 3 variables categóricas utilizadas
	prob_alrt_temprana	
	prob_auto_cura	
	lote	

Replica 12: Esta es una función creada para replicar (desde impala) los registros principales en n meses (en este caso cada obligación) de manera que se facilite el cruce histórico con las fuentes utilizadas y asi construir para cada variable ventanas de tiempo de 3 y 6 meses (esto limitado a la historia entregada).

La base de obligaciones es replicada 6 meses en base a los campos principales de: nit_enmascarado, num_oblig_enmascarado, num_oblig_orig_enmascarado y fecha_var_rpta_alt. Al cruzar con cada modulo se obtiene el valor de cada variable de 1 a 6 meses.

Nota: Se elimina el mes cero ya que este es el mes de observación de la obligación para la variable respuesta. Si este se considera posiblemente se estaría usando valores que al momento de procesar aun no existen

Fase 1: Data Engineering

Tendencias: Función para crear (desde impala) variables regresivas que permitan observar tendencias a N meses atrás de una variable seleccionada.

Al contar con el valor de cada variable de 1 a 6 se procede a ejecutar las tendencias de cada variable numérica donde resultan los valores de:

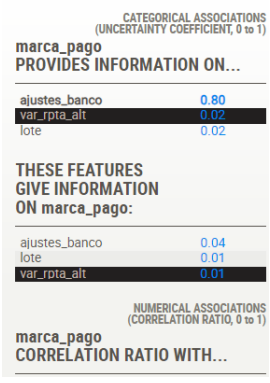
- variable a 1 mes
- promedio 3 y 6 meses
- desviación estándar (poblacional a 3 y 6 meses)
- valor máximo en 3 y 6 meses

dado lo anterior, por cada variable se obtienen 7 variables regresivas.

Dado el incremento de las variables, se necesita hacer una perfilación de los datos para establecer un punto de control y observar su comportamiento en base a: distribuciones, correlaciones, valores atípicos y posible aporte a la variable respuesta. Todo lo anterior se obtiene mediante **sweetviz**, el cual nos entrega un reporte html donde se puede hacer un análisis completo de cada variable como se observa a continuación:

- Análisis individual de cada variable, con su distribución, valores nulos y distintos

- Relaciones numéricas entre variables que permiten observar cuales pueden ser las variables candidatas a ser las mas influyentes en el modelo



Dado el análisis realizado de los reportes generados por cada base resulante de las tendencias se observó que:

- Las variables categóricas utilizadas son las únicas que mostraron ser significativamente relacionadas con la variable respuesta (según sweetviz)
- Se observo una gran cantidad de variables correlacionadas, sobre todo en la base de saldos diarios como se observa en su reporte

Nota: Todos los reportes generados están almacenados en la carpeta de one drive compartida.

Correlación: función creada para detectar una alta correlación entre 2 variables y seleccionar una entre ellas de acuerdo a un threshold dado como parámetro.

Luego de observar una alta correlación entre variables se ejecuta la selección con un valor de 0.8, es decir, si el índice de correlación entre dos variables es 0.8 o mas, se elimina una de estas de la base.

Fase 3: Modelado

Dada la exploración, construcción y selección de información previamente descrita se procede a preparar la base de entrenamiento.

Imputación: Dada la naturaleza de las variables y análisis realizado las variables numéricas no evidencian presencia de valores negativos, por tanto, se opta por imputar las variables con un valor representativo para la ausencia de este dato, estos se imputan con -99 (incluye también categóricas numéricas).

De otro lado las variables categóricas(texto) se imputan con un valor *no_data.*

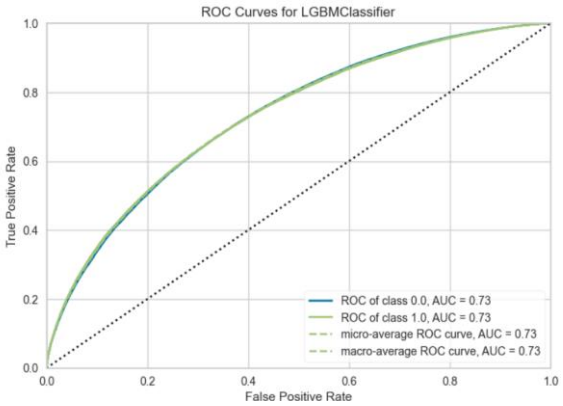
Normalización: Dada la diferencia de escala entre las variables seleccionadas, se utiliza la función MinMaxScaler de scikit learn

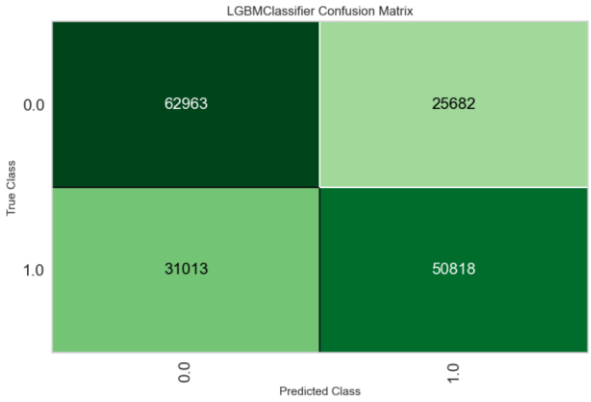
Outliers: Función creada para la detección de valores atípicos por el método de IQR y eliminar de la base de entrenamiento aquellos registros donde se detecten estos datos.

La ejecución de los experimentos en el entrenamiento del modelo consto de un proceso evolutivo, en el cual se parte de un modelo base y con cada ejecución se realizan ajustes a la base o parámetros con el objetivo de mejorar el performance del modelo. Estas ejecuciones se realizan mediante **pycaret** el cual facilita la comparación y tuneo de modelos.

Experimento 1: Modelo base solo con variables numéricas sin tendencias (solo vistas al mes anterior) ni variables categóricas.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.6667	0.7290	0.6207	0.6633	0.6413	0.3306	0.3313	1.6920
lr	Logistic Regression	0.6037	0.6402	0.5307	0.5983	0.5625	0.2027	0.2039	1.7440
svm	SVM - Linear Kernel	0.5755	0.6190	0.3837	0.5888	0.4644	0.1381	0.1469	1.5460



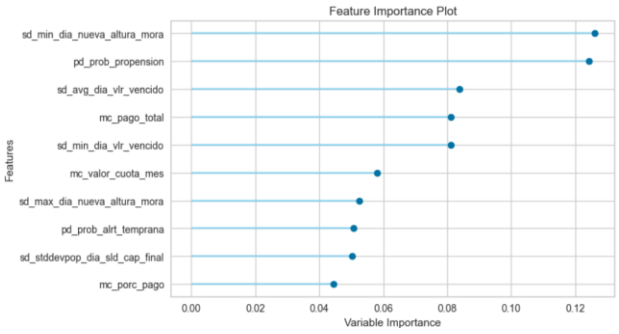


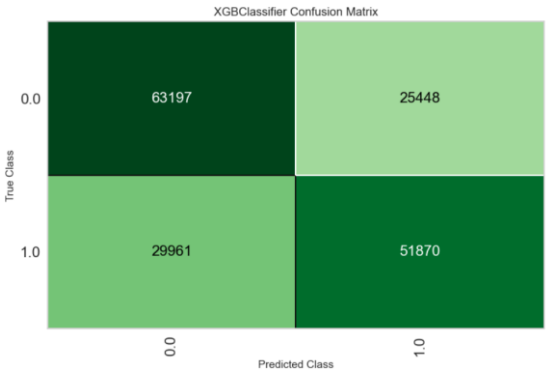
Para este experimento se consideran los algoritmos de ligthgbm, regresión lineal y máquinas de soporte vectorial donde el mejor modelo consiste en un **ligthgbm** con un F1 score de 0.64.

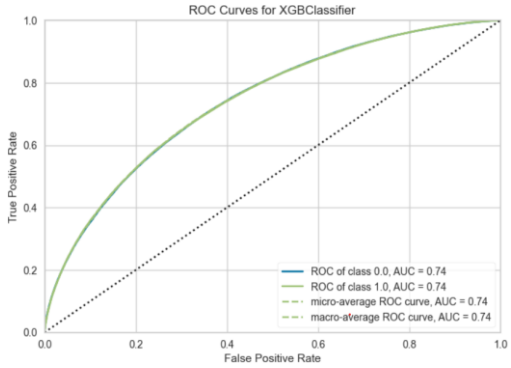
Experimento 2: Inclusión de algoritmo xgboost y tuneo de modelo

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
xgboost	Extreme Gradient Boosting	0.6732	0.7374	0.6309	0.6694	0.6496	0.3440	0.3445	1.7460
lightgbm	Light Gradient Boosting Machine	0.6667	0.7290	0.6207	0.6633	0.6413	0.3306	0.3313	1.7640
lr	Logistic Regression	0.6037	0.6402	0.5307	0.5983	0.5625	0.2027	0.2039	1.7380
svm	SVM - Linear Kernel	0.5755	0.6190	0.3837	0.5888	0.4644	0.1381	0.1469	1.3180

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.5086	0.7372	0.9943	0.4941	0.6602	0.0525	0.1502
1	0.5094	0.7392	0.9944	0.4945	0.6605	0.0540	0.1529
2	0.5099	0.7405	0.9943	0.4948	0.6608	0.0550	0.1544
3	0.5079	0.7371	0.9947	0.4938	0.6599	0.0513	0.1490
4	0.5081	0.7401	0.9942	0.4938	0.6599	0.0515	0.1483
Mean	0.5088	0.7388	0.9944	0.4942	0.6603	0.0529	0.1510
Std	0.0008	0.0014	0.0002	0.0004	0.0003	0.0014	0.0023





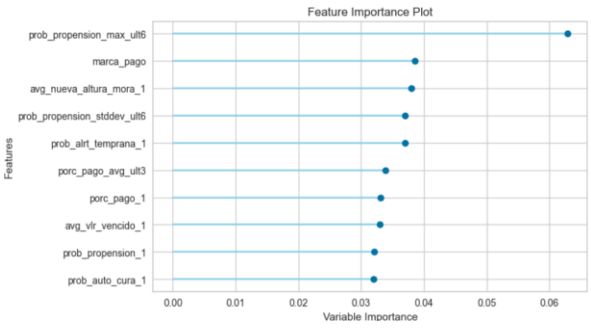


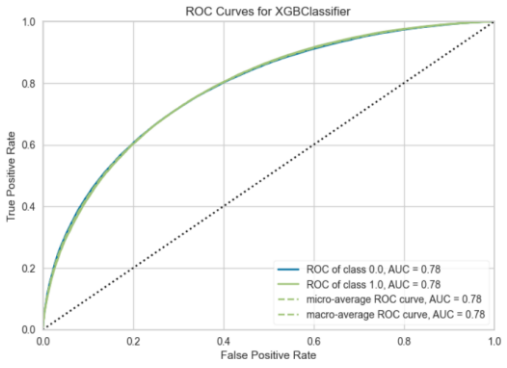
En este experimento se obtiene una mejora en el dempeño del modelo (F1=0.6469), manteniéndose estable luego de tunear este.

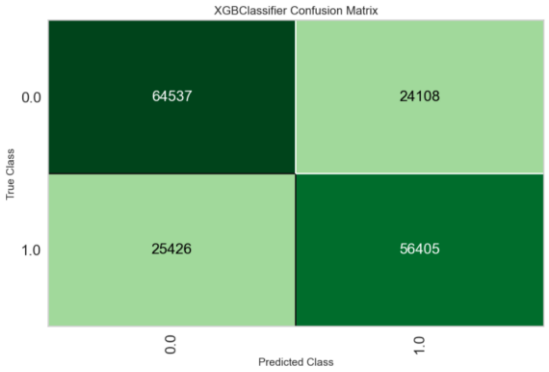
Experimento 3: Inclusión de variables tendencias, selección de variables por medio de la función de correlación y ejecución de los dos mejores modelos en los experimentos anteriores (lightbbm, xgboost)

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
xgboost	Extreme Gradient Boosting	0.6863	0.7551	0.6554	0.6796	0.6673	0.3706	0.3709	3.0380
lightgbm	Light Gradient Boosting Machine	0.6793	0.7461	0.6413	0.6745	0.6575	0.3563	0.3567	3.1120

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7059	0.7786	0.6822	0.6981	0.6901	0.4103	0.4104
1	0.7051	0.7780	0.6839	0.6963	0.6900	0.4088	0.4089
2	0.7062	0.7799	0.6800	0.6996	0.6896	0.4109	0.4110
3	0.7038	0.7770	0.6790	0.6964	0.6876	0.4062	0.4063
4	0.7033	0.7753	0.6794	0.6955	0.6873	0.4051	0.4052
Mean	0.7049	0.7778	0.6809	0.6972	0.6889	0.4083	0.4084
Std	0.0011	0.0016	0.0019	0.0015	0.0012	0.0023	0.0023







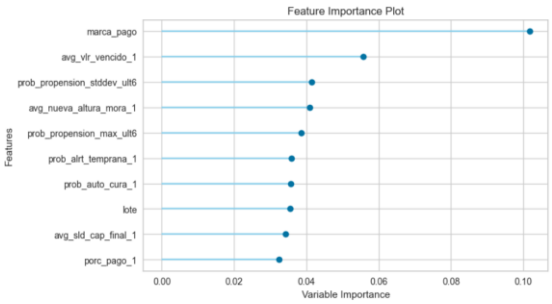
Para esta ejecución se sigue encontrando un mejor resultado con el algoritmo xgboost tuneado

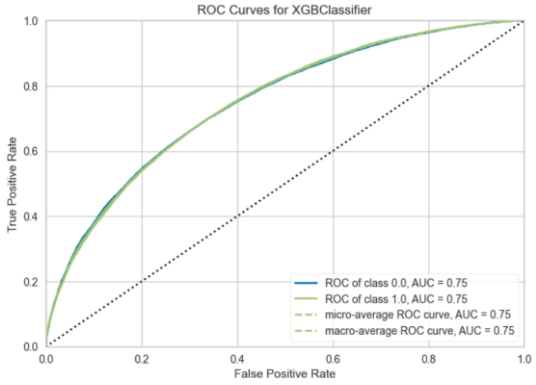
F1 = 0.6889

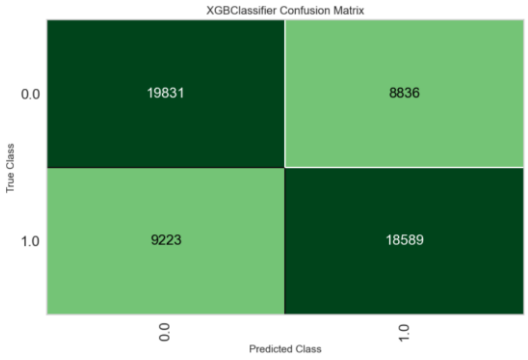
Experimento 4: Se incluyen las variables categóricas ajustes_banco y lote. Se eliminan los valores atípicos y solo se realiza ejecución de xgboost con un grid de parámetros personalizado en el tuning.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
xgboost	Extreme Gradient Boosting	0.6702	0.7356	0.6634	0.6657	0.6645	0.3402	0.3402	1.5880

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.6770	0.7437	0.6679	0.6734	0.6707	0.3537	0.3537
1	0.6722	0.7391	0.6626	0.6686	0.6656	0.3441	0.3441
2	0.6777	0.7445	0.6682	0.6744	0.6713	0.3552	0.3552
3	0.6830	0.7495	0.6740	0.6796	0.6768	0.3658	0.3659
4	0.6740	0.7414	0.6624	0.6712	0.6667	0.3476	0.3477
Mean	0.6768	0.7437	0.6670	0.6735	0.6702	0.3533	0.3533
Std	0.0037	0.0035	0.0043	0.0037	0.0039	0.0075	0.0075







Esta fue la ultima iteración ejecutada donde no se logra mejorar sustancialmente la métrica objetivo, por tanto, hasta la última ejecución se considera como un modelo no satisfactorio y se requiere hacer una exploración mas exhaustiva de parámetros, nuevas variables y suposiciones estadísticas.

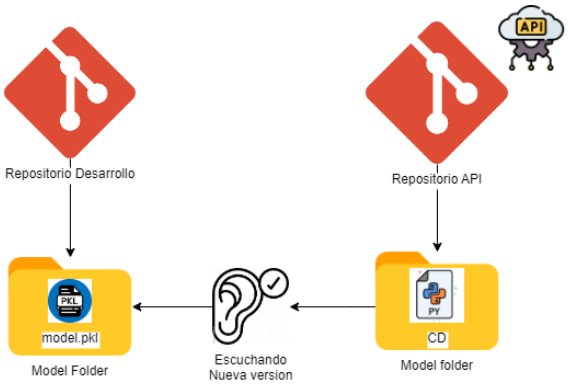
Monitoreo:

- **Test_inference:** función para testear la inferencia del modelo. El objetivo es realizar pruebas de comparación entre el valor de etiqueta original y la resultante del modelo. Para esto se recomienda periódicamente calcular la variable respuesta para hacer esta prueba.
- **Monitoring**: Funciones para el monitoreo de las variables donde se calcula tasa de variación, psi y desviación estándar que permiten analizar y controlar el data driff que puede afectar el performance del modelo.

CD:

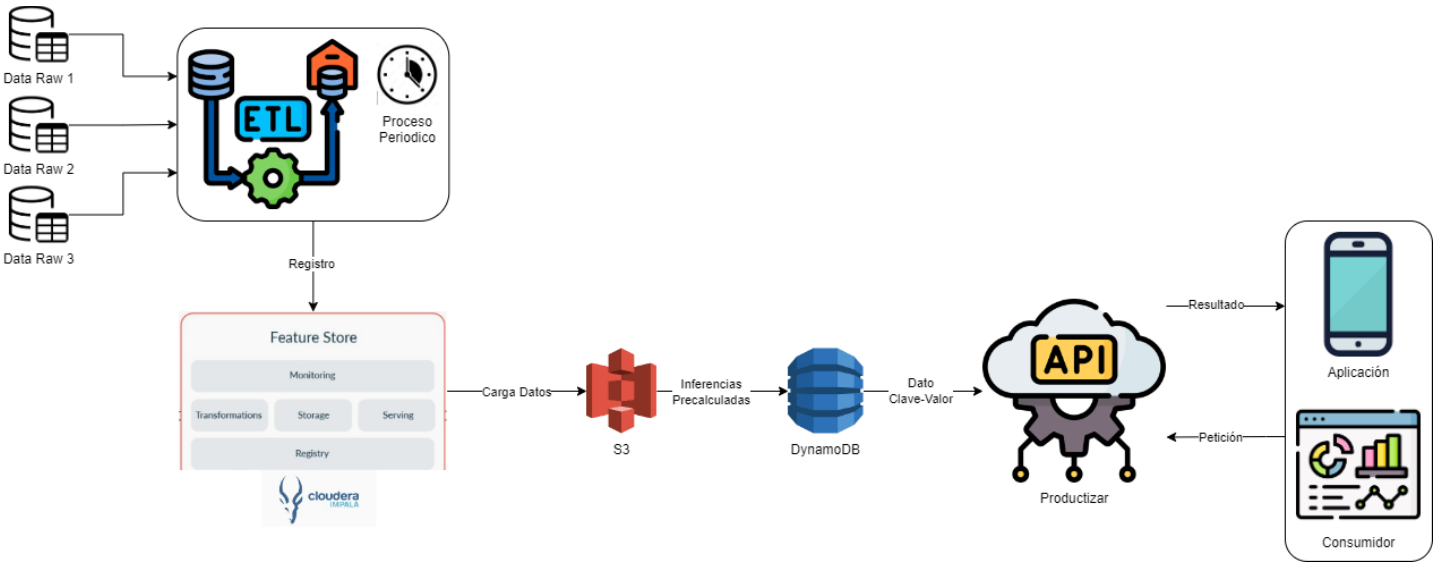
Dado que el modelo puede ser productizado y puesto en funcionamiento a través de un api, se desarrolla una estrategia que permite actualizar el modelo sin afectar la estructura de esta, lo que evita hacer un redespliegue.

Para esto el objeto modelo debe estar alojado en un repositorio independiente al de la api y ejecutar un proceso que este monitoreando las versiones de este hasta que detecte un cambio a causa de un reentrenamiento.



Este script se desarrolla utilizando la api Python de azure devops.

Diagrama Conceptual: Modelo como servicio:



Dado que el negocio requiere realizar la predicción con un mes de antelación el modelo seguirá una ejecución en batch, donde se plantea crear un api que se alimente de datos previamente calculados.

El flujo ejecución es el siguiente:

- Un proceso de etl ejecutado periódicamente para procesar los datos

- Un feature store para el registro, control y almacenamiento de transformaciones y las inferencias calculadas. Este aprovechando las capacidades de impala.
- Un servicio s3 para servir como transición de los datos que viajan a nube.
- Un servicio dynamo para almacenar las predicciones previamente calculadas.
- Una api desarrollada en fastapi para recibir y entregar las peticiones de los consumidores.