# Caso Práctico

## Enunciado

Una empresa de 15 trabajadores con sus sistemas albergados en la nube (ERP, CRM, control de *stock*…) quiere implantar una estrategia de analítica también en la nube (Azure).

Para ello desarrolla una serie de funciones *serverless* que se ejecutan periódicamente a las 2:00 am, y donde se copia la información de las bases de datos a ficheros CSV en un almacenamiento distribuido y se puede desde ahí leerlo con cualquier herramienta (*notebook*, diseñadores, AutoML…) e incluso enviar datos a aplicaciones de BI como Celonis y PowerBI.

En cuanto a sus necesidades de IA, necesitan un sistema predictivo para las ventas futuras, para así poder determinar el *stock* que necesitarán, otro que les prediga los clientes con una alta probabilidad de contratar un producto vía telefónica y, por último, varios algoritmos que se entrenan y se ejecutan de forma puntual para determinar determinados aspectos del negocio (como qué clientes tienen más propensión a aceptar una determinada oferta o un estudio especial de algún aspecto del negocio).

Considérese que el *dataset* que indica si un cliente va a contratar o no un producto vía telefónica es *bank marketing*, ubicado en <https://archive.ics.uci.edu/ml/datasets/Bank%2BMarketing>, y en concreto, “bank-full.csv”.

## Se pide

Contestar justificando todas y cada uno de los apartados y las decisiones tomadas:

**1 -** Describir la forma de productivizar cada modelo. ¿Qué tipo de herramienta se debería elegir para desarrollar cada modelo y por qué se ha elegido? ¿Cuál sería la forma de acceder a los modelos por parte de los usuarios? ¿Cada cuánto tendrían que reentrenarse?

**2 -** Describir el flujo de trabajo MLOps de los modelos. ¿Sería necesario en todos los modelos?

**3 -** ¿Qué cambios habría que hacer en los apartados anteriores si la empresa, en vez de 15 trabajadores, tuviese 10000?

**4 -** Productivizar el modelo de *bank marketing*. Hay dos opciones para elegir: mediante el diseñador de Azure o bien de forma manual en su propia máquina (serializando los modelos/transformaciones con Pickle).

1. Si el objetivo del modelo es lograr predecir previo a que se realice la llamada qué personas tienen una alta probabilidad de contratar, ¿se podría decir si hay alguna variable que no sea posible incluir en el modelo en producción?, ¿Cuáles son?
2. Probar varios modelos e ingeniería de características y dar con el modelo que mejor métrica presente.

Una vez elegido el mejor modelo, crear una llamada API para consultar el modelo. En caso de haber elegido hacer un modelo de forma manual, se tendrá que generar el código para responder a la llamada *GET* o *POST* con los nuevos datos. En caso de haber elegido hacerlo con Azure, se deberá crear el *endpoint* (hay que recordar que, posiblemente, al usar una cuenta gratuita, no se pueda ejecutar debido a que necesitaría más CPU de las que está permitido usar; no obstante, se deberá plantear la canalización y adjuntar captura de pantalla de ella aunque dé error).

## Describir la forma de productivizar cada modelo. ¿Qué tipo de herramienta se debería elegir para desarrollar cada modelo y por qué se ha elegido? ¿Cuál sería la forma de acceder a los modelos por parte de los usuarios? ¿Cada cuánto tendrían que reentrenarse?

Primero tendríamos que concretar que modelos queremos analizar, tras llegar a esa conclusión ya podemos responder a cada una de las preguntas, en este caso lo desglosamos en las 3 opciones:

### Sistema Predictivo de Ventas Futuras:

* **Herramienta de desarrollo**: Para el desarrollo de este modelo se recomienda Azure Machine Learning, ya que permite un modelado predictivo en la nube con capacidad de reentrenamiento y escalabilidad, además tras todo el módulo tenemos experiencia suficiente para trastear con él.
* **Forma de acceso**: Deberíamos crear un endpoint API que nos permita acceder a las predicciones desde cualquier sistema que necesite esos datos.
* **Frecuencia de reentrenamiento**: Podríamos aplicar una frecuencia mensual, ya que los datos de ventas suelen variar estacionalmente y aplicar este intervalo nos sería beneficioso para capturar patrones recientes.

### Modelo de Predicción de Clientes para Contratación Vía Telefónica:

* **Herramienta de desarrollo**: En el caso del modelo de predicción podríamos usar AutoML de nuevo en Azure Machine Learning, debido a su capacidad de explorar y comparar múltiples algoritmos y configuraciones de forma automática.
* **Forma de acceso**: De nuevo usaríamos un Endpoint API en este caso para predicciones en tiempo real antes de cada llamada telefónica.
* **Frecuencia de reentrenamiento**: Para la predicción de clientes usaríamos un intervalo trimestral y así nos podríamos asegurar de que los cambios en los patrones de contratación se reflejan en el modelo.

### Algoritmos puntuales para estudios específicos del negocio:

* **Herramienta de desarrollo**: En el caso de los algoritmos puntuales podríamos hacerlo a través de Notebooks de Azure ML o incluso en scripts locales usando Python y librerías como ya hemos visto (como Scikit-Learn), ya que al ser un entrenamiento puntual, los modelos no requieren un reentrenamiento muy frecuentemente.
* **Forma de acceso**: Podríamos llevarlo a cabo a través de accesos API temporales o incluso con reportes.
* **Frecuencia de reentrenamiento**: En este caso concreto solo llevaremos el reentrenamiento cuando se necesiten nuevos estudios o análisis puntuales.

## Describir el flujo de trabajo MLOps de los modelos. ¿Sería necesario en todos los modelos?

Queremos diseñar un flujo de trabajo MLOps que sea adecuado y útil para cada modelo y tras esto comprobar si es necesario implementarlo en todos

### Flujo general de MLOps:

* En el flujo incluimos pasos como puede ser la preparación de los datos, la experimentación, el entrenamiento de modelos, la validación, implementación y por último la monitorización. Si quisiéramos aplicarlo a modelos que requieran una actualización regular como puede ser las ventas futuras o incluso la contratación vía telefónica deberíamos usar o Azure DevOps o utilizar pipelines en Azure ML.

### ¿Es necesario en todos los modelos?:

* **Modelos de Ventas futuras y Contratación vía telefónica**: Estos modelos requieren un reentrenamiento y también llevar a cabo monitorización constante para que mantenga la precisión, por lo que si es necesario en este tipo de modelos.
* **Modelos Algoritmos Puntuales:** Este tipo de modelos son entrenados de forma esporádica solo cuando se lleva a cabo un nuevo estudio o análisis, por lo que no es necesario tampoco una monitorización contante, solo en esos momentos de actualización, por lo que no sería necesario implementar un flujo completo de MLOps.

## ¿Qué cambios habría que hacer en los apartados anteriores si la empresa, en vez de 15 trabajadores, tuviese 10000?

Tendríamos que analizar primero como cambiaria nuestra estrategia si la empresa fuese de tales dimensiones

### Escalabilidad de la infraestructura

* Tendríamos que usar clústeres de procesamiento más grandes en Azure para manejar ese incremento en la demanda. No solo conlleva que sean más grandes los clústeres, sino que tendríamos que implementar más nodos y CPU en las instancias que se ejecuten para así mejorar su rendimiento y reducir los tiempos de procesamiento.

### Automatización y monitorización avanzada

* tendríamos también que mejorar la monitorización con el uso de herramientas más avanzadas como puede ser Azure Monitor que nos sería útil para vigilar el rendimiento de los modelos en tiempo real. También a la hora de automatización destacaríamos mejoras como puede ser en el proceso de reentrenamiento donde ejecutaríamos automáticamente cuando en los datos se detecten cambios importantes.

### Acceso a modelos

* De forma similar a la manera inicial usando accesos API escalables pero tendríamos que hacer modificaciones a nivel de permisos de acceso, ya que si aumenta tanto la cantidad de empleados tendríamos que otorgar permisos por departamentos para así tener un control mejor y una mayor seguridad

## Productivizar el modelo de *bank marketing*. Hay dos opciones para elegir: mediante el diseñador de Azure o bien de forma manual en su propia máquina (serializando los modelos/transformaciones con Pickle)

### Si el objetivo del modelo es lograr predecir previo a que se realice la llamada qué personas tienen una alta probabilidad de contratar, ¿se podría decir si hay alguna variable que no sea posible incluir en el modelo en producción?, ¿Cuáles son?

### Probar varios modelos e ingeniería de características y dar con el modelo que mejor métrica presente

### Una vez elegido el mejor modelo, crear una llamada API para consultar el modelo. En caso de haber elegido hacer un modelo de forma manual, se tendrá que generar el código para responder a la llamada *GET* o *POST* con los nuevos datos. En caso de haber elegido hacerlo con Azure, se deberá crear el *endpoint* (hay que recordar que, posiblemente, al usar una cuenta gratuita, no se pueda ejecutar debido a que necesitaría más CPU de las que está permitido usar; no obstante, se deberá plantear la canalización y adjuntar captura de pantalla de ella aunque dé error).

Lo primero que tendremos que hacer será dirigirnos a Azure machine learning, abrir la máquina que hemos creado y utilizado para todos los ejercicios anteriores. Abrimos Azure Studio, nos vamos a la sección de datos e introducimos una nueva fuente de datos, subimos el CSV y lo nombramos como casoPracticoBanco. Tras esto nos dirigimos a la sección de ML automatizado y creamos un nuevo ML automatizado:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Seleccionamos el tipo de tarea que va a llevar a cabo, en nuestro caso Clasificación ya que queremos que el modelo determine una de dos clases que serían contrata o no contrata. También seleccionamos en este panel la fuente de datos deseada:Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Tras seleccionar la tarea y la fuente nos vamos a la configuración de estas, en las que podemos seleccionar la columna destino que será la columna y (String), también la opción de Aprendizaje profundo que en este caso no sería necesario activarlo ya que no es un modelo muy complejo. También se nos otorga la posibilidad de establecer límites a nivel de ejecución para controlar los recursos utilizados y el tiempo de ejecución:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Establezco una serie de limites ya que al tener la versión gratuita ya comprobé con los casos prácticos anteriores que se satura la maquina y no logro obtener los resultados deseados.

Tabla

Descripción generada automáticamente con confianza media

Seleccionamos el clúster de proceso donde se ejecutará y lo enviamos:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Tras ejecutarlo obtenemos varios resultados Interfaz de usuario gráfica, Aplicación, Correo electrónico

Descripción generada automáticamente

Como se puede observar en la imagen se ha deducido que el mejor modelo es VotingEnsemble con un AUC ponderado de 0.93287, tras esto podemos revisar el resto de las métricas para saber realmente el rendimiento del modelo, clicando sobre Ver todas las demás métricas:

Tabla

Descripción generada automáticamente con confianza media

Tras analizar su rendimiento con todas las métricas procedemos a registrar el modelo nombrándolo Modelo\_VotingEnsemble, dando sobre el botón de Registrar Modelo: Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Si nos dirigimos a modelos se puede visualizar el modelo registrado

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza mediaSeleccionamos el modelo que deseamos ver los detalles y dentro de este se nos muestra este botón, que nos servirá para crear el Endpoint en tiempo real, tras clicarlo tendremos que configurarlo:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

A continuación se muestra la configuración de este, originalmente el recuento de instancias estaba en 3 por determinado, pero superaba la cuota que tenía asignada con mi cuenta:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

El problema que con la suscripción actual gratuita da lugar a conflictos y no deja crear ese punto de conexión, aunque la canalización sí que está realizada pero al dar conflictos con mi suscripción gratuita no puedo comprobar su funcionamiento.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

4.1 Variables:

Para asegurar que el modelo funcione correctamente en un entorno de producción, es necesario que analicemos las variables del dataset y descartar aquellas que no sean útiles. Las variables que identificó como inadecuadas son las siguientes:

* + Duration: Representa la duración de las llamadas telefónicas (En segundos). Este dato solo lo conocemos una vez que la llamada ha terminado por lo que no estaría disponible antes de que la interacción con un cliente ocurra. El objetivo que tenemos con el modelo es predecir la probabilidad de que un cliente acepte una oferta antes de realizar la llamada, por ello duration no es útil ya que no sirve para una predicción anticipada.
  + Contact: Nos indica el tipo de contacto realizado con el cliente, en algunos casos este campo tiene valores como puede ser “unknown” lo cual nos indica una falta de información. En un entorno de producción si contamos con valores desconocidos puede reducir la fiabilidad lo que nos afectaría a la precisión del modelo.
  + Poutcome: Esta variable representa el resultado de una campaña de markenting anterior, mostrándonos si un cliente participó o no, esta variable puede sernos útil en muchos contextos, pero en producción puede introducirnos sesgos ya que no todos los clientes deben tener un historial en campañas anteriores. Este sesgo puede hacer que el modelo favorezca a clientes que han interaccionado con nosotros en otras campañas, lo cual no queremos que ocurra ya que buscamos predecir la probabilidad de conversión de clientes NUEVOS.

Esas son las variables principales que pienso que deberíamos excluir en el modelo de producción para garantizar predicciones precisas basadas en datos disponibles en tiempo real y útiles para una campaña sin introducir sesgos o dependencias con información incompleta o desconocida.

4.3 API:

Vamos a llevarlo a cabo de forma manual utilizando la librería Pickle para así serializar el modelo en mi maquina y poder crear una API sencilla en Python ya que con la suscripción actual como ya se ha observado da muchos problemas la creación de un endpoint.

Lo primero creamos un notebook para meter un script de entrenamiento para el modelo:

Interfaz de usuario gráfica, Aplicación, Correo electrónico

Descripción generada automáticamente

Aplicamos RandomForestClassifier en scikit-learn y este no puede trabajar directamente con datos categóricos. Para solucionar esto convertimos las columnas a un formato numérico antes de entrenar el modelo a través de usar la técnica de One-Hot encoding y así logramos que se ejecute de forma correcta el notebook:

Texto

Descripción generada automáticamente

Tras ejecutar el notebook ya tendremos el modelo entrenado y guardado en formato Pickle, ahora procedamos a la creación de la API, creamos un archivo llamado app.py e introducimos el código que nos permitirá cargar el modelo y procesará las solicitudes de predicción en tiempo real:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Aquí se muestra el código del archivo app.py que es la API en Flask lo que como ya he mencionado nos permitirá cargar el modelo y las solicitudes, este archivo tiene que estar en el mismo directorio que bank\_model.pk:

Texto, Aplicación

Descripción generada automáticamente

Tras esto solo queda comprobar el funcionamiento de la API por ello iniciamos un servidor Flask en modo de desarrollo que se ejecuta en <http://127.0.0.1:5000> como se puede observar en el código. En la siguiente imagen se ve como está el servidor Flask en ejecución.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Tras iniciar el servidor simplemente debemos comprobar que funciona la API esto lo podemos hacer a través de una solicitud POST con un conjunto de datos de prueba, en la propia consola donde está el servidor iniciado o en otro archivo .py, en mi caso generé otro archivo para que sea más claro de ver:

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Al ejecutarlo obtenemos el resultado de la predicción:



Esto nos indiciaria que el modelo predice que el cliente no contratará el producto con una probabilidad del 4%.