**Prueba técnica Gestor SR con énfasis en IA.**

**Teoría:**

1. En la empresa GA, en el área de compras necesitan CLASIFICAR y organizar los correos que llegan a la bandeja de entrada entre 4 tipos de correos (Compras cementos, Compras energía, Compras concretos y correos generales o de otra índole). Esta tarea se le encomienda a usted, como es el Gestor SR en temas de analítica e IA puede solicitar al área interesada los recursos humanos que necesite para llevar a cabo este proyecto, también puede solicitar en tecnología todo lo que necesite, además tiene las bandejas de entrada de correos históricos de los analistas que reciben estas solicitudes con aproximadamente: 5500 correos de compras cementos, 2700 correos de compras de energía, 1100 correos de compras concretos y 12876 correos generales o de otra índole.

**Explique cómo resolvería este problema, metodología, algoritmos, modelos, arquitectura del proyecto etc.**

En principio este es ejercicio de minería de textos o de procesamiento de lenguaje natural, donde el conocimiento se adquiere adoptando bases de datos de textos. Varios estudios previos plantean que este tipo de estudio con los correos electrónico podrían estudiarse desde dos enfoques:

Una es la categorización de texto por machine learning empleando modelos de clasificación como KNN, Naive Bayesiano Supervisado o vectores de soporte SVM y random forest, arboles de decisión e incluso, redes neuronales.

La otra es la coincidencia de patrones de texto y planillas. Una forma es la creación de un diccionario con palabras que tienen posibilidad de aparecer en determinado correo y se categorizan. Otra forma, y de manera más reciente, hay modelos basados en Deep learning que implementen una red neuronal basada en modelo lstm para clasificar correos.

Entonces, la forma de trabajo sería, un etiquetado manual de correos, una exploración y comparación de correos (plantillas) y formas que puedan llegar, preprocesamiento de la data, pruebas de machine learning para construcción de un clasificador automático con alguno de los modelos mencionados anteriormente.

La propuesta metodológica y su arquitectura de una manera muy general, está planteado de la siguiente forma:

1. Se debe recoger la data, como son archivos de correo, esto se debe almacenar en HTML, y se deberían retirar las etiquetas para quedar solo con contenido textual.
2. Preprocesamiento, limpieza de datos, transformación, reducción de la dimensionalidad
3. Aprendizaje, Python usando modelos SVM, XGboost, etc. De ser muy grande la data, se sugeriría usar spark.
4. Evaluación y selección del modelo
5. Ejecución
6. La arquitectura, hay varios elementos, primero se debe tener guardada la información en un espacio, como un datalake, databricks y un ftp.

Desconozco la forma, pero posiblemente se pueda llevar a cabo en azure, puntualmente en ML Ops, También se podría agruparlo en un Docker para llevarlo a despliegue en una appp, web, o front

1. Seis meses después de haber desplegado un modelo de regresión en producción, los usuarios se dan cuenta que las predicciones que este está dando no son tan acertadas, se le encarga a usted como Gestor SR en temas de IA que revise que puede estar sucediendo.

**¿Cree que el modelo esté sufriendo Drift?**

**¿Cómo puede validarlo?**

**¿De ser así, que haría usted para corregir esto?**

**Explique sus respuestas.**

Si, el modelo puede estar perdiendo predicción dado que esta dejando de lado los eventos y situaciones que vienen ocurriendo en los últimos 6 meses que afectan el comportamiento de la predicción, por tanto, es importante tener un tiempo de reentramiento más corto para que estos efectos no sigan ocurriendo. La forma de evidenciar si está o no perdiendo efectividad es calculando las predicciones versus la realidad y calcular el valor de algunas métricas de acierto de predicción como el MAPE, el RMSE o el R2, y compararlas con las que dieron en la última puesta en producción en su entrenamiento, si es evidente su caída, podemos asumir que el modelo perdió asertividad.

Como lo mencioné anteriormente, una de las formas de asegurar que esto no ocurra, es reentrenando los modelos con menor tiempo, para así posibilitar que los modelos aprendan lo que sucedió recientemente y calculen un pronóstico mucho más semejante a la realidad, puesto que es probable que ocurriesen eventos de datos atípicos, tendencias, estaciones o incluso situaciones geopolíticas.

Si es muy difícil poder lograr que se reentrene por los altos costos, se debería hacer una especie de alerta que muestre el nivel de acierto que va teniendo semana a semana y poder haciendo ajustes que no requieran modelación (por ejemplo, la suma de datos atípicos, etc).

1. Su equipo de trabajo está trabajando en un chatbot con generación de texto utilizando el modelo GPT-3.5, según cómo funciona este modelo, ¿cómo haría usted para hacer que las respuestas del chatbot estén siempre relacionadas a conseguir cierta información particular del usuario y no empiece a generar texto aleatorio sobre cualquier tema?

**Explique su respuesta.**

**Practica:**

Summa-sci requiere construir un servicio (Backend) para una aplicación de experiencia del colaborador, esta aplicación lo que hace es recibir en la interfaz de usuario (Frontend) unos parámetros de una lista desplegable ingresados por el usuario y con esto un valor en números que corresponde al pronóstico de la demanda de las compras de la compañía Cementos Argos, asociado a esos parámetros.

La información es enviada desde la interfaz hacía el servicio, este la recibe en formato json y utiliza estos datos para generar una clasificación (Alpha, Beta) que son códigos que le indican al analista que tipo de compras de materiales debe hacer, y en que cantidades, finalmente el servicio debe responderle a la interfaz de usuario con un json en el que se especifica cual es el código.

Hay 3 archivo útiles para realizar este proyecto:

dataset\_demand\_acumulate.csv

Este archivo contiene la información de la demanda entre el 2017-01 hasta el 2022-04 siendo (año-mes).

dataset\_alpha\_betha.csv

Este archivo contiene la información de todas las variables involucradas para realizar la clasificación de si un registro es Alpha o Betha, este cuenta con más de 7000 registros.

to\_predict.csv

Este archivo es lo que buscamos predecir, cuenta con 3 registros los cuales ya tienen toda la información completa, excepto la demanda y la clase.

Para este proyecto usted deberá:

1. Deberá crear un modelo que pueda pronosticar la demanda de 2022-05, 06 y 07, de Cementos Argos. El código de esto deberá hacerlo en un Jupyter notebook, el output debe ser un archivo con todos los datos incluyendo los pronósticos, además de un gráfico donde se pueda apreciar cual fue su data de entrenamiento, de validación y pronóstico, dentro de este gráfico se debe apreciar las métricas con las que decidió que su modelo estaría dando un resultado confiable (Puede usar las que considere).



Ilustración 1. Ejemplo visualización

1. Deberá entrenar un modelo de clasificación para generar las respuestas que hará el servicio, el código deberá hacerlo en un Jupyter notebook, el output de esta ejecución será un archivo con el modelo entrenado y un txt con las métricas generadas por este, también indicaciones de que modelo usó, cantidad de datos de entrenamiento, testeo, etc. Es libre de hacer reducción de dimensionalidad, escalado, aumento de data, balanceo de data o cualquier técnica que mejore su pronóstico. (Tenga en cuenta que si redujo la cantidad de predictoras para el momento de hacer el pronóstico debe hacerlo también a esos datos)
2. Deberá crear un proyecto en el cual exponga un api (Puede usar flask, FastApi, django, o cualquier librería para exponer apis) y sea ejecutable desde un archivo main.py, este proyecto deberá consumir el modelo que creó en el punto 2, recibir el json de una solicitud post, realizar la clasificación y entregar la respuesta. El entregable será el repositorio del proyecto y unas instrucciones para ejecutarlo, con archivo readme y requirements. Tanto el json del post como el del output usted puede organizarlos como quiera, solo necesita que tenga las variables que el modelo usa
3. Complete el archivo “to\_predict.csv” con la información de la demanda que pronosticó en el punto 1. Y realice las 3 peticiones al servicio que diseñó, finalmente entregue un archivo con los resultados que generó su servicio con el modelo predictivo.
4. **(Opcional) De ser posible, cree un contenedor en Docker con la aplicación, así puede ser ejecutada y desplegada en ambiente productivo de manera más sencilla, deberá entregar el Docker file (Docker compose), para que nuestro equipo haga el build y ejecute este contenedor. (Opcional)**
5. Los entregables de este proyecto deben subirse a su repositorio de github, trate de hacer buen manejo de las ramas, commits, pr, etc, y también buen manejo de las carpetas dentro del repositorio, el archivo de respuestas teóricas debe ser subido en formato pdf con el nombre “teoria\_.pdf”

Se debe determinar la frecuencia en que se van a entrenar los modelos para mantenimiento y operación, al igual que se debe considerar la latencia y rendimiento. Luego se debe poner a disposición los datos de entrada y los modelos para que el sistema tome los datos y haga las predicciones y las entregue para su publicación en un front o dashboard.

Existen varios ciclos con el cual se podría poner en productivo. Uno de ellos y el de mayor auge, es el uso de tecnologías en la nube principalmente en la de Azure que se realizarían en servidores remotos, empleando el servicio de ML Ops para el despliegue y puesta en producción.

Entonces, esta solución se aborda desde el flujo de ML ops. Primero se debe hacer la canalización de compilación, en dónde compila el código y se ejecuta un conjunto de pruebas. Acá se crea el área de trabajo que hospeda los recursos, el proceso de entrenamiento, el script de aprendizaje y todas las conexiones con las fuentes de datos. Luego el paso de canalización de reentramiento, en donde se entrena el modelo, se evalúa, registra y ejecuta el modelo según lo programado o cuando haya nueva data y, por último, una canalización de versión para cuando haya nueva data.

Otra forma de realizar el despliegue es en local y es el dispositivo el que se programa cada cierto tiempo para que corra los modelos, esto es más manual, y no tiene tanta capacidad de computo como un servicio de nube, y no entregaría predicciones “en vivo” sino cuando termine de correr. Herramientas para esto son Docker, Flask y FastAPI, para empaquetar el modelo como una API y acceder localmente desde el navegador.