Teoría

Punto 1

El proyecto debe abordarse en varias fases, la primera sería todo lo que es el entendimiento a profundidad del problema, lo que ayudaría a entender diferentes temas y tener todo el contexto de la necesidad, posteriormente preprocesamiento de los datos donde se entienden los datos y al final arroja como resultado la extracción de las características que se necesitan, aquí es esencial abortar temas como la tokenizacion de los datos, lo cual separará los datos de cada correo en partes, la eliminación de stop-words, como puntos, comas o conectores, lematización el cual ayudará a llevar a cada palabra resultante a su estado base, finalmente vectorizar los datos en su forma numérica, aplicando un one hot encoding por ejemplo.
En la siguiente fase, se debe elegir el modelo a usar, no sin antes haber realizado el split de los datos en entrenamiento y testeo. De acuerdo con esto, se eligen algoritmos clásicos como SMV o random forest, o también el modelo puede utilizar algoritmos más enfocados en NLP, como BERT, RoBERTA, VADER, los cuales son más avanzados en estos temas y vectorizan los datos para clasificarlos. Con estos últimos hay que tener en cuenta que algunos son más efectivos en inglés, por ello es que hay que hacer un preprocesado de datos y traducirlos de ser seleccionados. Adicionalmente podrían usarse estos 3 mencionados y de acuerdo a la clasificación elegir la que más tenga, es decir, si dos clasificaron compras cementos y uno compras de energía, la decisión final por mayoría seria compras cementos
Además, al hacer la clasificación elegir la métrica de evaluación como precisión, recall o F1-Score.
Finalmente crear el pipeline de todo el proceso para que todo el proceso sea igual en recepción del correo, procesamiento, clasificación y almacenamiento.

Por otra parte, pensando en la infraestructura:

- Frontend: Interfaz de usuario para seguimiento y uso de la solución

- Backend: proceso que soporte temas como integración al servidor de correos, el almacenamiento(base de datos) de los datos históricos como la recepción de los nuevos y la clasificación del modelo

Recursos:

- Servicios de nube Azure
- Herramientas de seguimiento del proyecto

Por último, establecer una periodicidad para el seguimiento y revisión de la solución.

Punto 2

Si, evidentemente el mundo está en constante cambio, sea cual sea el tema que esté usando un modelo de predicción hay que hacer un constante monitoreo de estos. Ya que el entorno puede estar cambiando, y los datos de entrada con el tiempo estén evolucionando en sus tendencias lo que hace que el modelo con el tiempo pierda su precisión. Además, puede suceder que haya cambios en los datos, como en las escalas o en las métricas, por ejemplo un dato venia en kilómetros cuando se entrenó el modelo, y ahora está llegando en millas, o que si un modelo usaba el rango del salario de las personas, es normal que con los años estos rangos suban cambiando el panorama.

Ahora bien, para detectar si un modelo está sufriendo de drifting, existen varias pruebas estadísticas, inicialmente se puede graficar una distribución de los datos históricos con los que se entrenó el modelo y los nuevos datos, a veces visualmente se es posible detectar anomalías cuando son muy evidentes, con esto comparar medidas básicas de tendencia central como media o varianzas.

Por otra parte, cuando no es tan evidente y con esto no fue concluyente, se puede aplicar pruebas estadísticas como el K-S test, este es basado en la comparación de las distribuciones, cuya hipótesis es que ambos conjuntos de datos son iguales, si se rechaza se dice que hay una desviación del modelo.

También está IEP o índice de estabilidad de la población, la cual compara una misma variable en los dos conjuntos de datos, el con que se entrenó el modelo y el actual y determina que tanto se ha movido la distribución

Asi mismo, hay una librería en Python llamada evidently, la cual es usada para monitorear la salud del modelo y arroja cambios en su eficiencia.

En conclusión, hay que entrar a validar varias cosas en los datos, haría diferentes pruebas y llegado el caso de obtener resultados que confirmen el drifting, reentrenaría el modelo.

Punto 3

Para que el chatbot no se disperse en una conversación aleatoria, hay que definir parámetros iniciales que si o si el chatbot debe obtener para ayudar al usuario con la solución. La limitación de los inputs bien sea con carruseles, botones o respuesta rápidas prediseñadas para que el usuario conteste alguna de las opciones esperadas, puede ser una gran opción. Además estos elementos de control ayudarían a aumentar la velocidad de iteración y experiencia del usuario y no estar a merced de cualquier input por parte del usuario. Además de esto, en este caso como se usa el modelo de ChatGPT, seria importante proveer todo el contexto del tema que se requiera, si se tienen datos con los que se pueda darle a entender todo el contexto para que pueda entenderlo y estar ajustado siempre al tema requerido y no empezar a hablar de temas complemente desconocidos sino que sus respuestas usen nuestros datos para responder, para esto se debe usar una base de datos de vectores y con esto puede darle contexto a chatGTP.