Tarea 2 – Ingeniería de Software – Magíster en Ciencias de Datos UC

Nombre del estudiante: Luciano Davico

1. Con respecto a cada uno de los cuatro valores ágiles:

- **¿Cómo crees tú que se manifestó ese valor en el proyecto y por qué? Es decir, ¿se valoró más el ítem a la izquierda o el ítem a la derecha, y en qué se notó esta mayor valoración?**
- **¿Crees tú que el ítem más valorado (el de la izquierda o derecha) resultó ser el ítem "correcto", desde el punto de vista de haber sido un aporte al proyecto? Justifica**

Respuesta

Valor ágil 1, personas e interacciones sobre procesos y herramientas

Este valor ágil se manifestó en el proyecto, ya que si bien se utilizó una tecnología especializada para desarrollar entrenamiento e inferencia automática como Sagemaker, en el que hay muchos servicios inmersos para cada caso de uso, se privilegió mantener el uso de esta herramienta de forma simple, evitando ser muy sofisticados y que sea flexible para el usuario que utilizaría el software. En este sentido se valoró más el ítem de la izquierda *personas e interacciones.* Sí creo que es el ítem correcto, dado que de lo contrario, si hubiésemos preferido utilizar al máximo el servicio Cloud en primera instancia, la curva de aprendizaje para lograr una implementación correcta y un uso del framework sería mayor por parte del usuario y pasaríamos a resolver "no-problemas", por lo que sería incluso contraproducente en términos de productividad.

Valor ágil 2, software que funciona sobre documentación completa

Este valor se manifestó de forma explícita, dado que en los comienzos del proyecto se comenzó con una prueba de concepto, la cual fue siendo refinida en base a MVPs cada vez. De esta forma, siempre se privilegió que el entregable fuera un programa que funcione y sea mostrable todos los días viernes, siendo así más valorado el ítem de la izquierda *software que funciona*. Tan así fue, que incluso se dejó la documentación del software hacia el final, dado que podría cambiar mucho en el transcurso del proyecto. Creo que dadas las características del software, en el que la tecnología utilizada no era conocida de antemano, la decisión correcta fue privilegiar el ítem de la izquierda, sino de lo contrario hubiésemos puesto esfuerzos en vano al documentar sobre incertidumbre.

Valor ágil 3, colaboración con el cliente sobre negociación de contratos

Este valor ágil también se manifestó explícitamente en el proyecto, dado que ninguno de los que participamos tenía un conocimiento concreto del servicio Cloud que se iba a utilizar, por lo que no disponíamos de información perfecta antes de comenzar el proyecto. De esta forma, se privilegió el ítem de la izquierda y de conversar constantemente con los stakeholders del proyecto, siendo realistas en el backlog que podíamos cumplir para los plazos pactados y adecuándonos siempre a la necesidad del usuario target que iba a operar con este framework. Sí creo que es el ítem correcto dado que, al existir la incertidumbre de cómo funcionaba a priori el servicio, era necesario ir conversando con el cliente sobre cómo extraer valor de la herramienta, en vez de negociar desde nuestro punto y querer implementarla a toda costa.

Valor ágil 4, responder al cambio sobre seguir un plan

Este valor se manifestó durante el proyecto más cercanamente en la mitad de la ejecución, dado que en principio se fijó un plan con objetivos concretos y un backlog inicial, pero durante el tiempo nos fuimos dando cuenta que debíamos revisitar ciertos flujos y piezas de código del software que estábamos desarrollando. De tal forma, tuvimos reuniones en que se presentó el estado del arte del proyecto y fuimos dejando más de lado un plan, y más bien fuimos respondiendo a priorizar la usabilidad del framework para el usuario final. Diría que desde que empezamos a valorar más el ítem de la izquierda *responder al cambio* comenzamos directamente a agregar más valor para los stakeholders, dado que en principio nos apegamos mucho al plan y nos comenzamos a alejar de las necesidades de negocio específicas que resolvía el nuevo proyecto.

2. “Llena” el backlog del producto con unas 10 funcionalidades – una frase breve que describa qué se quiere agregar, o la actualización que se quiere hacer – y prioriza justificadamente estas funcionalidades.

Respuesta

1. Leer documentación de los componentes que interactúan en la arquitectura de Sagemaker
2. Definir orquestador de los procesos de AutoML de forma automática y periódica
3. Crear una clase base de entrenamiento e inferencia que implemente métodos comunes a existir en todo el código de ML para tener un diseño de código más robusto.
4. Establecer entrenamiento de varios modelos automáticamente con un mismo dataset
5. Crear un pipeline de despliegue automático de imágenes de Docker en AWS
6. Definir una tarea de aprobación manual para elegir el modelo entrenado con mejores métricas
7. Crear tests para el entrenamiento e inferencia automática
8. Documentar brevemente como utilizar el framework en repositorio de código

9. Crear archivo de configuración del servidor de inferencia con parámetros respecto a la capacidad del servidor.
10. Crear un *flag* que decida si realizar optimización de hiperparámetros para cada modelo

Como se puede ver, cada funcionalidad esta priorizada, siendo la tarea 1 la más prioritaria y la tarea 10 la menos prioritaria. Lo anterior se da porque primero era necesario entender cómo funcionaba la herramienta Sagemaker, entender si realmente nos podía resolver o aportar valor al negocio, respecto a disminuir esfuerzos en la productivización de modelos. Este paso es crucial y puede determinar el nivel de esfuerzo de las demás tareas. Luego de aquello, se consideró que debía existir automatización de los procesos, dado que ese era justamente el foco del proyecto y, en conjunto con lo anterior, el hecho de poder entrenar simultáneamente varios modelos y de forma automática era justamente el caso de uso principal a considerar. Luego de eso, también era importante entender cómo el modelo estaba siendo entrenado y también tener la posibilidad de elegir el modelo entrenado que más se acomode al negocio, acorde a las métricas (muy importante dependiendo del caso de uso). Las tareas siguientes, desde la 7 en adelante, se priorizaron menos, dado que son funcionalidades "deseables" del proyecto y que eran alcanzables en el corto plazo, pero que se podía vivir sin ellas de momento, dado que el framework permitía tomar decisiones de forma manual para la elección del modelo y de los dataset. Sin embargo igual eran funcionalidades importantes.

3. Planifica el primer sprint de tu proyecto:

- Elige justificadamente una duración de sprint, entre una semana y un mes
- Elige justificadamente tres o cuatro funcionalidades del backlog del producto para el primer sprint – el *backlog del sprint*
- "Ejecuta" el primer sprint, tanto como sea posible a base de lo que realmente ocurrió en el proyecto; en particular:
    - ¿cuál podría haber sido el resultado de una de las reuniones de Scrum Diario?
    - ¿cuál podría haber sido el resultado de la revisión del Sprint?
    - ¿cuál podría haber sido el resultado de la Retrospectiva del Sprint?

Respuesta

- Duración del sprint: dos semanas durante los primeros dos meses y luego una semana. Se consideró esto dado que se estimó que en las primeras semanas se experimentaría una curva de aprendizaje más alta del servicio Sagemaker, por lo que en un sprint de una semana no habría mucho valor entregado para un mvp. Luego de dos meses ya se podría tener un avance concreto con las principales funcionalidades, por lo que después serían sprints de una semana para afinar detalles específicos o tener nuevas funcionalidades bien concretas

- Tareas elegidas: **(1), (2), (3), (4).** Se eligieron estas tareas, dado que las primeras dos son más de diseño y se requería en primer lugar un diseño robusto. Se eligió la tarea **(3)** dado que para cumplir el entrenamiento de varios modelos, existirían métodos comunes en el entrenamiento e inferencia y por lo tanto se hacía imperativo tener un diseño de código en base a herencia y desacoplar las lógicas correspondientes a cada modelo de forma simple, evitando repetir código. Luego, la tarea **(4)** es la principal funcionalidad que se quiere lograr, lo cual es posible después de tener un código desacoplado, provisto por el diseño de herencia en la tarea anterior.

- En el resultado de las primeras 3 reuniones de Scrum Diario no hubo muchos accionables, sino más bien bloqueantes, dado que estábamos en proceso de revisar documentación oficial de Sagemaker, la cual no es clara y no existían ejemplos de implementación para todos los casos de uso. La curva de aprendizaje se tomó una semana completa del sprint, por lo que partimos atrasados en un principio. Dado lo anterior, el resultado de la revisión del sprint probablemente no hubiera sido muy satisfactorio, dado que no se tendría un MVP, pese a que es esperable que esto ocurriera. De esa forma ocurrió y los stakeholders lo entendieron completamente, considerando incluso que es necesario primero entender bien los límites del servicio que implementaríamos y por lo tanto, vale la pena tomarse un buen espacio de lectura de documentación y creación de pruebas de concepto. Dado lo anterior, en el resultado de la retrospectiva podría revisitarse la planificación de los diferentes sprints por venir y considerar mejorar el proceso de estimación de esfuerzo de las tareas, a modo de darle mayor prioridad y holgura a entender como funciona Sagemaker, antes de comenzar a implementar el caso de uso a ciegas.