Autor: Dimitrio Mandamadiotis Kalfagiannis CI 23723158

Tutor: Yaneth Moreno

**DESARROLLO DE UN SISTEMA PARA INFERIR LA DESERCIÓN DE LOS CLIENTES EN UN E-COMMERCE**

**CAPITULO 3**

**Análisis, Planificación y Diseño del Sistema**

En este capitulo se explicaran los procesos que fueron necesarios para la creación del sistema para inferir la deserción de clientes. Esta es la fase inicial de construcción del backlog donde se analizan los objetivos del proyecto y se definen las lineas de diseño base para la creación del sistema.

**3.2 Arquitectura del Sistema**

La solución al problema planteado consta de un sistema donde se integran tres módulos: el modelo, la API y la aplicación web (Figura 1). El módulo de modelo se encarga de procesar la data que se obtuvo en la plataforma Kaggle y generar un modelo estadístico para la predicción de la tasa de deserción de los clientes. Este proceso se realiza una vez, es decir, no es necesario crear otro modelo o ajustar el modelo actual con los datos ingresados por un usuario mediante la interfaz.

Figura 1: Arquitectura del Sistema

La API sirve de middleware o capa intermedia entre el modelo y la interfaz del usuario, es decir, esta se encargaría de recibir los datos de la aplicación web, procesarlos haciendo uso del modelo y devolver el resultado de nuevo a la interfaz para que el usuario pueda verlos.

Finalmente, la interfaz tiene como finalidad permitir introducir al usuario al sistema, recibir su entrada como una lista de transacciones que están representadas por una fecha y un identificador de cliente, enviar esa información a la API para ser procesada, recibir el resultado y mostrarle un gráfico de probabilidad por cada cliente donde se exprese la probabilidad de que este haya desertado en un punto en el tiempo futuro.

La comunicación entre el modulo del modelo y la API se realizaría mediante un archivo PKL que contiene los parámetros del modelo creado, el cual será importado por la API al momento de su ejecución y utilizado para generar predicciones. La API y la aplicación web hablarían mediante peticiones HTTP e intercambiarían la información en formato JSON.

**3.1 Requerimientos**

A pesar de que la metodología de desarrollo es ágil, se consideró importante definir una serie de requerimientos como base que debe cumplir el sistema, esto con la finalidad de mantener un enfoque durante la implementación y que las tareas del backlog tengan un fin.

Acorde a lo descrito en la sección anterior en la cual se desarrolló una arquitectura para el sistema, la lista de requerimientos por módulo que se estableció es la siguiente:

**3.1.1 Modelo**

**3.1.1.1 Requerimientos Funcionales**

* El sistema genera un modelo estadístico que pueda predecir la deserción de clientes de un e-commerce.

**3.1.1.2 Requerimientos no Funcionales**

* Se debe importar las transacciones de un archivo CSV.
* Se debe procesar la data en un dataframe con las características definidas por la biblioteca lifetimes.
* El modelo se debe guardar en un archivo con formato PKL en la carpeta del módulo API.
* El código fuente del modelo debe estar disponible dentro del repositorio del proyecto en GitHub.
* Se debe implementar el modelo en un Jupyter Notebook.
* En el Jupyter Notebook debe estar expresado el proceso de generación del modelo.
* Se deben mostrar ejemplos de uso del modelo.

**3.1.2 API**

**3.1.2.1 Requerimientos Funcionales**

* Crear una ruta (/conditional-probability-alive) donde se reciba una lista de transacciones por usuario y se retorne la probabilidad de vida de cada cliente en 500 unidades de tiempo desde la primera transacción.

**3.1.2.2 Requerimientos No Funcionales**

* La API será usada por una aplicación web.
* La API usará la arquitectura REST.
* El formato de entrada y salida debe estar descrito en el archivo README.md correspondiente a la carpeta del proyecto.
* Las instrucciones de ejecución deben estar contempladas en el archivo README.md correspondiente a la carpeta del proyecto.
* El formato debe especificar en cada campo con su clave y tipo de dato aceptado.

* Permitir el acceso libre a la API.
* La API debe procesar y retornar datos en formato JSON.
* La API debe estar disponible de manera pública mediante un enlace de prueba.
* El código fuente de la API debe estar disponible dentro del repositorio del proyecto en GitHub.

**3.1.3 Aplicación Web**

**3.1.3.1 Requerimientos Funcionales**

* El usuario ingresa la lista de transacciones de los clientes y obtiene la probabilidad de que los clientes desertara de su negocio en un grafico lineal.

**3.1.3.2 Requerimientos No Funcionales**

* Se debe validar que las fechas ingresadas por el usuario sigan el formato DD-MM-AAAA.
* No se pueden ingresar o enviar transacciones sin fecha o identificador de cliente.
* El identificador de cliente es una cadena de caracteres con tamaño mayor de tres.
* Las instrucciones de ejecución deben estar contempladas en el archivo README.md correspondiente a la carpeta del proyecto.

* Se debe describir brevemente las instrucciones de uso en el archivo README.md correspondiente a la carpeta del proyecto.
* La aplicación web debe procesar y retornar datos en formato JSON.
* Se debe describir brevemente las instrucciones de uso la interfaz.
* Se debe acceder a la API mediante un método POST.
* El código fuente de la aplicación web debe estar disponible dentro del repositorio del proyecto en GitHub.

**3.2 Diseño de Interfaz de Usuario**

...

**3.3 Priorización**

…