UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS

FACULTAD DE INGENIERÍA

CARRERA DE CIENCIAS DE LA COMPUTACIÓN

PROGRAMACIÓN CONCURRENTE Y DISTRIBUIDA

TRABAJO FINAL

ALUMNO N°01: José María Rojas (u201623243)

ALUMNO N°02: Renzo Ravelli (u201616446)

ALUMNO N°03: Sergio Sugahara (u201513410)

Link al Repositorio de GitHub: https://github.com/SergioSugaharaC/PCD.git

Link del video a YouTube: https://youtu.be/eP-Q0WoGnr4

Lima, 2020-2

Introducción:

El golf es un deporte al aire libre que es afectado seriamente por el estado del campo y el clima, como humedad del césped que genera fricción al momento de que la pelota bote y ruede, o el viento que, dependiendo de la velocidad, cambia la trayectoria del tiro. Por ello, tanto los deportistas como los que proporcionan estas áreas para practicar el deporte deben validar que se pueda llevar a cabo un juego de acuerdo al clima.

Descripción del Problema:

El problema que queremos abarcar con el presente trabajo es brindar apoyo a un gestor del deporte para poder determinar si es que bajo ciertas condiciones climatológicas se puede llevar a cabo un partido de Golf sin que estas condiciones puedan afectar de manera muy negativa el flujo del juego.

Motivación:

Creemos que es un buen escenario para experimentar con este modelo y hacerlo más robusto, de tal manera que mientras se consigue más datos y precisión, se pueda escalar esta propuesta a otros tipos de deportes o eventos.

Objetivos:

El objetivo principal a cumplir con la realización de este trabajo, es la implementación de un algoritmo de Machine Learning que se ejecute de manera paralela y distribuida con una cantidad de 5 nodos conectados que pueda determinar si un partido de Golf se puede jugar o no en base a características climatológicas.

Diseño (Arquitectura o Componentes):

Descripción del dataset utilizado

La data utilizada para el proyecto es una versión reducida del dataset Weather Nominal. Este dataset consta de reportes climatológicos para determinar si es posible o no realizar un juego de golf en base a distintas condiciones. Estas condiciones (features) son 4 atributos del clima y la quinta columna es la clase en la que se indica si es que se puede o no llevar a cabo el partido.

Los campos considerados fueron:

- **Outlook:** Overcast, Rainy, Sunny

- **Temperature:** Hot, Mild, Cool
- **Humidity:** High, Normal
- **Windy:** True, False
- **Play:** Yes, No

Descripción del algoritmo utilizado

El algoritmo de machine learning empleado para la realización de este trabajo fue Naïve Bayes, el cual considera el teorema de Bayes y algunas hipótesis de simplificación para calcular las probabilidades de que una tupla sea de una determinada clase en base a sus características.

Desarrollo:

El trabajo es desarrollar una API cuyo backend sea el algoritmo de machine learning y corra mediante consultas JSON. El algoritmo de machine learning seleccionado fue Naive Bayes debido a la naturaleza de los datos y el problema que es una clasificación simple por probabilidades, o al menos en los posibles primeros experimentos. El backend ha sido desarrollado en GO (Golang) aplicando distribución y funcionando en 5 nodos con funciones principales del algoritmo.

El primer proceso es la lectura del dataset (CSV separado por “,”) de un repositorio remoto usando una URL. Al ser el primer proceso, recibe solicitudes sencillas de administrar, como el inicio de una nueva predicción; además se implementa una espera a que haya una respuesta correcta del dataset, sino sería vuelta a probar la solicitud. En futuras implementaciones, se podría aplicar un primer filtro de normalización de la data, pero en nuestras experimentaciones, por motivos prácticos, se usó el dataset ya normalizado y sin algún dato corrupto.

El segundo proceso es un evento concurrente que se encarga de contar las veces que reincide un evento dentro de todos los escenarios planteados. Este proceso no se lleva a cabo si el conjunto de datos no ha sido leído aún. En el presente problema, nuestra clasificación de variables nominales es simple y rápida.

El tercer proceso es el cálculo de las probabilidades de acuerdo a la recurrencia de los eventos contados previamente. Este proceso tiene un tiempo de espera de aproximadamente de 10 milisegundos al proceso de conteo para evitar un conteo no completo y ofrecer un resultado final incorrecto.

El cuarto proceso es el clasificador usando un arreglo inexistente en el dataset con el cual determinará las predicciones para los resultados de cuándo sí se puede jugar y no se puede jugar. Ambos cálculos son ejecutados de manera paralela usando los comandos nativos de GO.

Finalmente, el quinto proceso es la impresión de los resultados obtenidos y el modelo listo para evaluar las situaciones deseadas. Las consultas por el usuario sería mediante un postman en JSON con los cuatro atributos usados para la predicción y se le daría la respuesta casi inmediatamente. Asimismo, al estar el modelo ya entrenado para el usuario, se permitiría que el Nodo 1, que incluye el proceso de leer el dataset, pueda volver a ser llamado por el mismo usuario para otro experimento, lo que conlleva a reajustar las probabilidades.

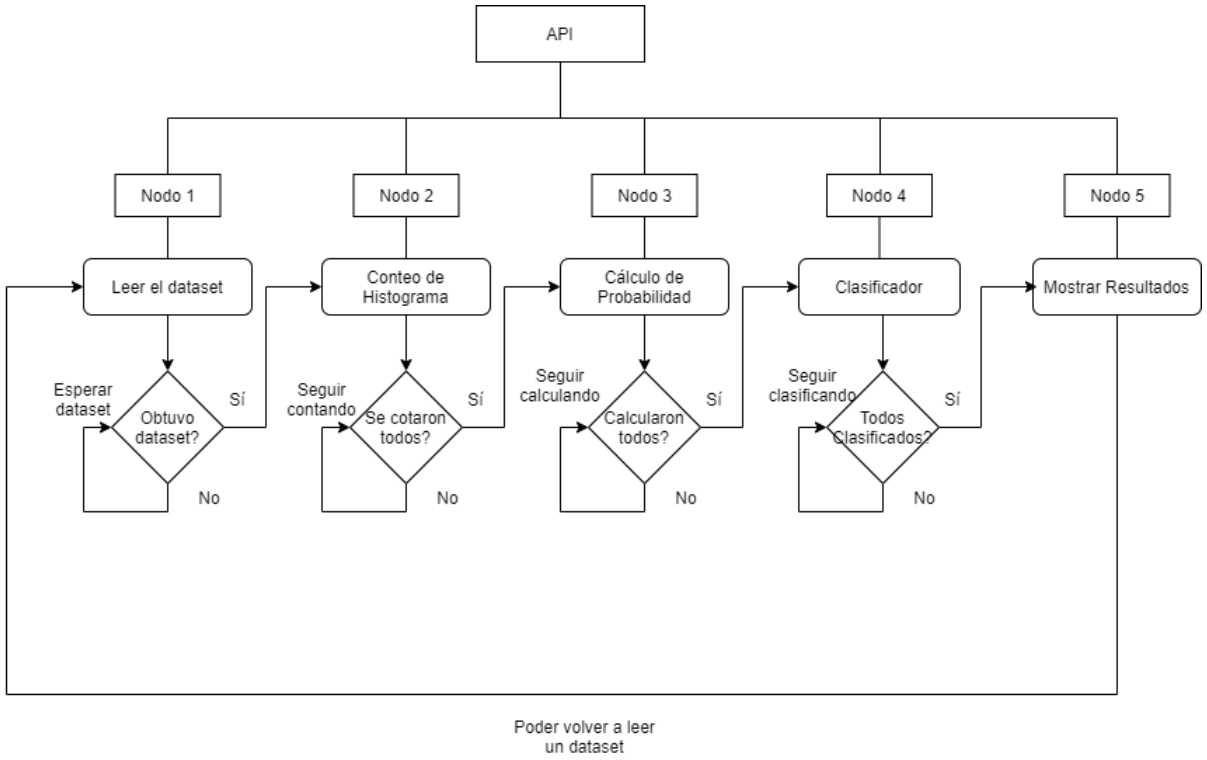

Figura 1: Diagrama de funcionamiento de la API (Fuente: Elaboración Propia)

Conclusiones:

Luego del desarrollo del proyecto presente, se pudo llegar a la conclusión que si bien el algoritmo utilizado es simple y la naturaleza de sus datos también, el resultado que logra es

interesante para aquellos interesados tanto como jugadores como para los organizadores de eventos relacionados a este deporte.

Para el gestor del deporte, el uso de una herramienta que usa machine learning le puede beneficiar para maximizar sus ofertas para su público, ya que se encargaría de organizar los juegos solo cuando estos sean viables desde el aspecto climático.

De manera similar, para los estudiantes de ciencias de la computación, el usar concurrencia permite aprovechar los recursos del procesador al momento de realizar trabajos al momento de repartir en distintos hilos en contraste con saturar uno solo. Asimismo, el trabajo distribuído reduce la carga de trabajo que tiene que hacer cada equipo (por practicidad del trabajo, se realizó con distintos puertos del mismo host local, pero en una mayor proyección la optimización será mayor).