Pontificia Universidad
JAVERIANA
Cali

PROYECTO 2 (Spark Streaming + Kafka + MLlib)

Presentado por:
Juan Camilo Hernandez Saavedra
Julian Paredes Conde
Augustin Valencia

PROCESAMIENTO DE GRANDES VOLÚMENES DE DATOS

CALI, OCTUBRE DE 2021

- **Contexto:**

  Este conjunto de datos refleja los incidentes de delitos denunciados (con la excepción de los asesinatos) que ocurrieron en la ciudad de Chicago desde 2001 hasta el 2017. Los datos se extraen del sistema CLEAR (Análisis e informes de aplicación de la ley ciudadana) del Departamento de Policía de Chicago. Para proteger la privacidad de las víctimas de delitos, las direcciones se muestran solo a nivel de bloque y no se identifican ubicaciones específicas. El dataset se encuentra dividido en 4 archivos .csv donde cada archivo contiene información en un rango de 3 años, a excepción del último archivo que contiene información en un rango de 4 años (2012-2017). Para nuestro caso de estudio se trabajará con el último archivo pues consideramos que este cuenta con la cantidad de información que creemos justa para el estudio.

- **Estructura del Dataset:**

  Para esta entrega se va a utilizar el dataset ya limpio que se obtuvo de la entrega pasada. El dataset cuenta con 15 atributos de los cuales podemos encontrar:

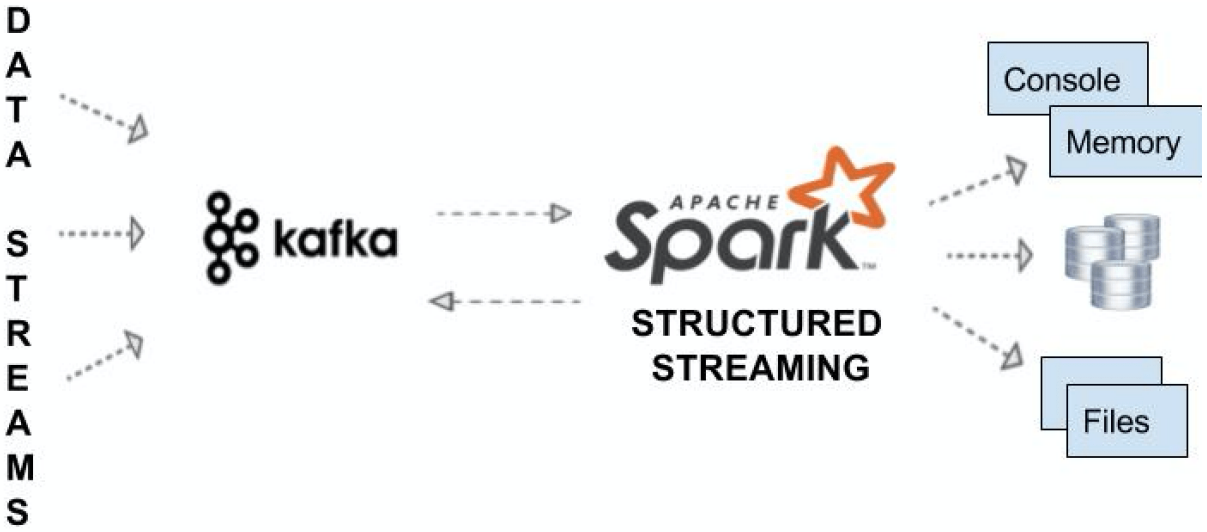
```
root
 |-- Arrest: integer (nullable = true)
 |-- Domestic: integer (nullable = true)
 |-- Beat: integer (nullable = true)
 |-- District: double (nullable = true)
 |-- Community Area: double (nullable = true)
 |-- X Coordinate: double (nullable = true)
 |-- Y Coordinate: double (nullable = true)
 |-- IUCR_index: double (nullable = true)
 |-- Location Description_index: double (nullable = true)
 |-- FBI Code_index: double (nullable = true)
 |-- Block_index: double (nullable = true)
 |-- mesDel: integer (nullable = true)
 |-- diaDel: integer (nullable = true)
 |-- horaDel: integer (nullable = true)
 |-- minutoDel: integer (nullable = true)
```

- **¿Qué se va a predecir?:**

  Continuando con la primera entrega, se busca predecir si dado un delito en cierto mes se va a realizar un arresto o no. Todo esto con base en la información relacionada con la demografía de los delitos cometidos en cierto mes específico. Esto es un problema de clasificación binaria, pues las salidas de los modelos estarán representados de manera binaria, donde 1 significa que se realizará un arresto y 0 significa que no se realizará un arresto.

  A diferencia de la primera entrega, los datos que se van a utilizar para entrenar y mantener actualizado el modelo de machine learning, van a llegar constantemente por medio del data Streaming. Todo esto con el fin de simular la llegada constante de delitos en un mes específico para que el modelo de machine learning se actualice con nueva información cada cierto tiempo.

- **Tecnologías y Arquitectura:**



  - Para simular el envío constante de información, se crea un **script en python** con el objetivo de cargar los delitos que sólo ocurren en cierto mes específico, para posteriormente ser enviados al servicio de Kafka. El script se conecta al servicio de Kafka por medio de la librería Kafka-python.

    En la estructura general de la arquitectura el script haría de producer, configurado de forma que se envíen batches de 120 filas cada 120 segundos. Esto se debe a que el script hace fetch de una fila cada 1 segundo y es enviado al buffer a la espera de que se cumplan los 120 segundos mencionados anteriormente, todo esto con el objetivo de tener cierta sincronía con el consumer. Las filas son transformadas a formato JSON y codificadas a UTF-8 para ser enviadas.

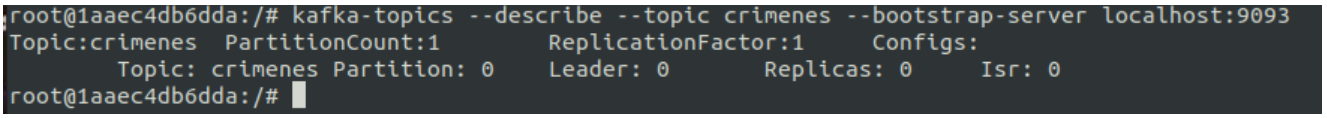
```
class Producer:
    def __init__(self, _topic, _freq, _ruta):
        self.__topic = _topic
        self.__freq = _freq if isinstance(_freq, int) else int(_freq)
        self.__producer = KafkaProducer(bootstrap_servers='localhost:9093',
                                         value_serializer=lambda x: json.dumps(x).encode('utf-8'),
                                         batch_size = 1638400, buffer_memory = 33554432, linger_ms= 500000)
        self.__gestor = gestorArchivos(_ruta)
        self.__datos = None

    def cargarDatos(self, _mes):
        self.__datos = self.__gestor.cargarArchivos(_mes)

    def cambiarDatos(self, _ruta, _mes):
        self.__gestor.setRuta(_ruta)
        self.cargarDatos(_mes)

    def enviarDatos(self):
        for index, valor in self.__datos.iterrows():
            dict_data = dict(valor)
            self.__producer.send(self.__topic, value=dict_data)
            print(f'Message {index + 1}: {dict_data}')
            time.sleep(self.__freq)
```

- Para transmitir los datos, se va a utilizar la tecnología de **Kafka,** Esta se encuentra configurada con un topic de nombre crímenes con una sola partición, la cual va a servir para almacenar los datos que se van a enviar por streaming al consumer de Spark.



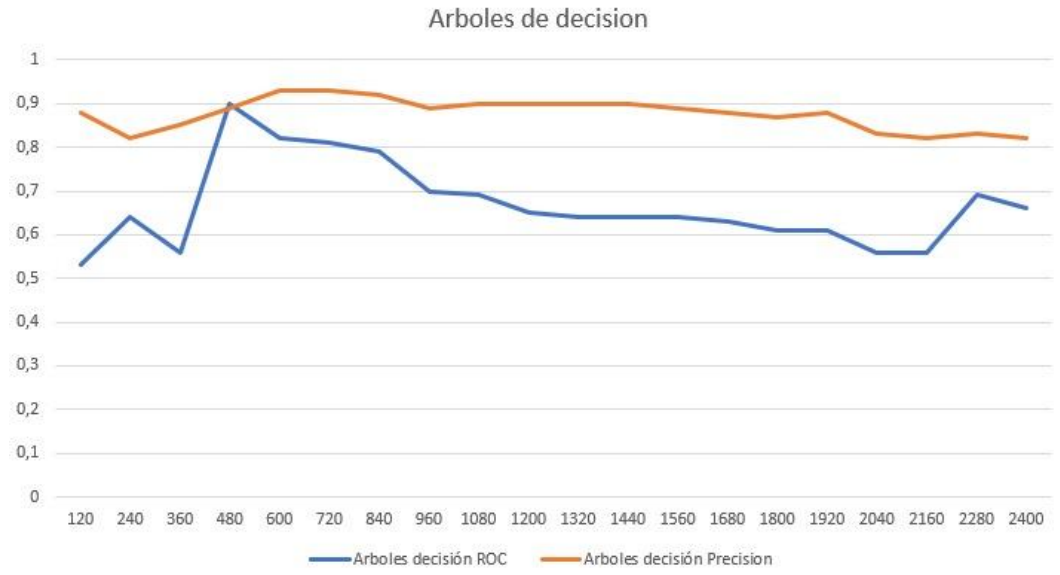
- Para el procesamiento y creación de los modelos de machine learning se va a utilizar **Apache Spark.** Para nuestro caso, nuestro cluster de Spark cuenta con un nodo master y dos nodos que funcionan como workers. En la estructura general de la arquitectura el cluster de Spark servirá como consumidor de la información que proviene de Kafka.

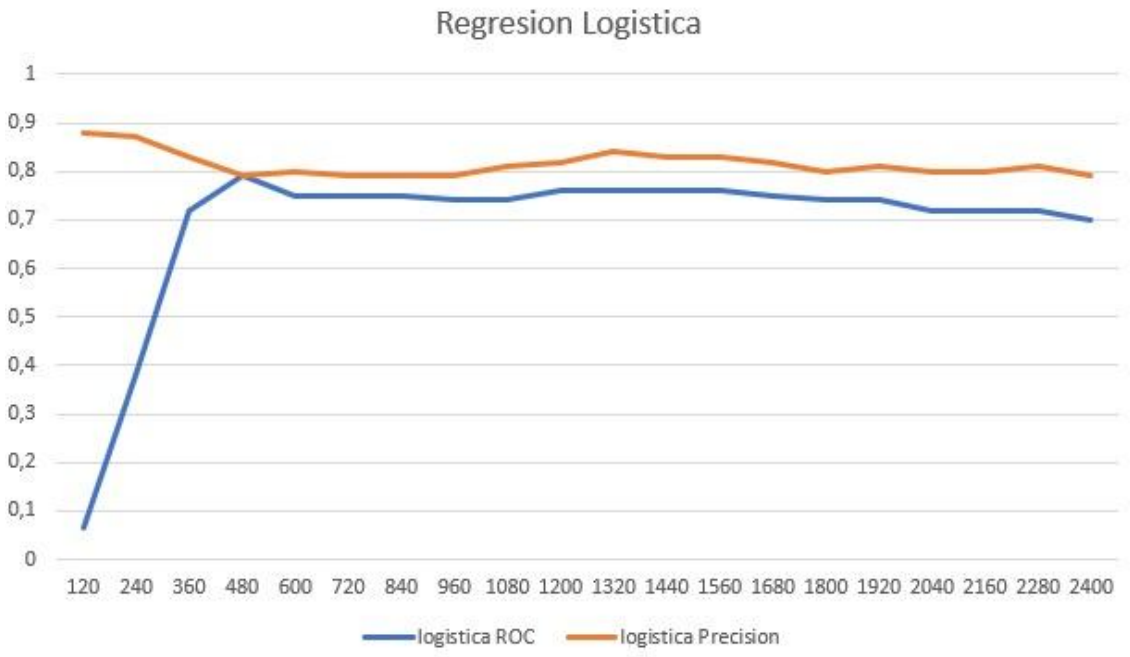
  Se espera que se haga fetch de los datos ubicados en el topic de crímenes cada segundo por ende, una vez se lean todos los datos la cola de mensajes de Kafka quedará vacía a la espera de otro batch, esto con el fin de no saturar el servicio de comunicación y Spark.

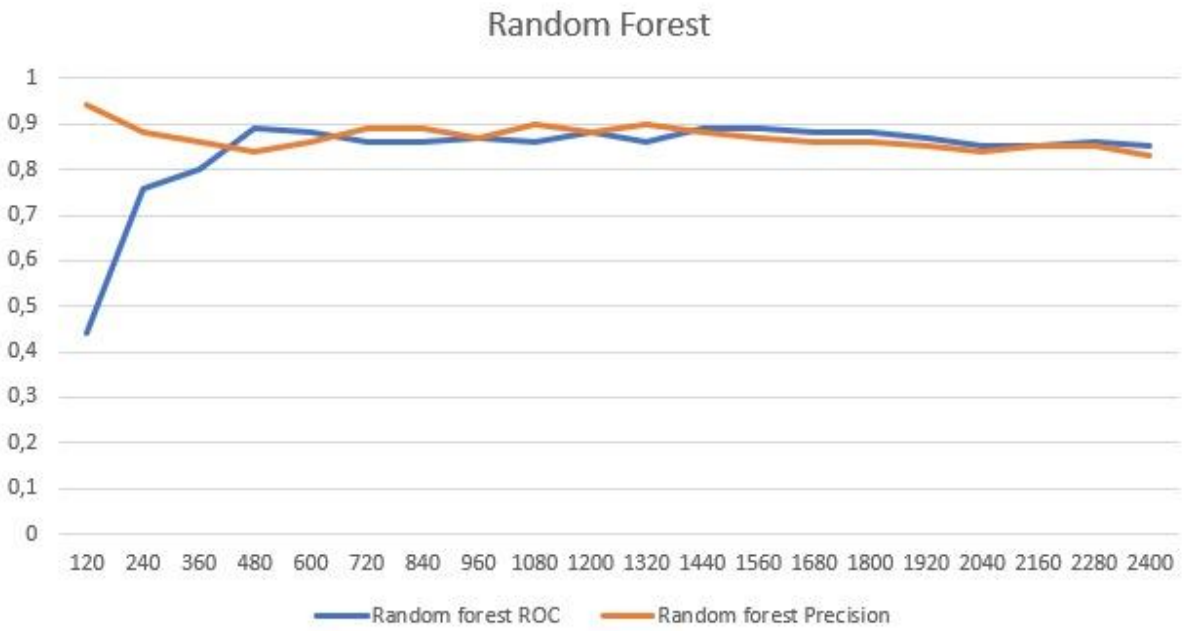
- **Implementación Algoritmos Machine Learning (MLlib):**

  El proceso que se utilizó para entrenar y probar los modelos de machine learning, consiste en que a medida de que se reciba un batch de información este es utilizado para actualizar los modelos constantemente. Cada dos minutos el Producer envía los batches con el tamaño que se describió anteriormente, una vez obtenida la nueva información, esta se une con los batches anteriormente para posteriormente reentrenar los modelos y obtener sus métricas.

  Al implementar la estrategia se puede observar como las métricas de los modelos cambian a medida que se actualizan con nueva información. Para resumir todas las métricas obtenidas a través del tiempo se realizó una gráfica para cada modelo en donde se puede visualizar las qué métricas se obtuvieron con cada batch.


Arboles de decision
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
120 240 360 480 600 720 840 960 1080 1200 1320 1440 1560 1680 1800 1920 2040 2160 2280 2400
Arboles decisión ROC
Arboles decisión Precision


Regresion Logistica
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
120 240 360 480 600 720 840 960 1080 1200 1320 1440 1560 1680 1800 1920 2040 2160 2280 2400
logistica ROC
logistica Precision


Random Forest
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
120 240 360 480 600 720 840 960 1080 1200 1320 1440 1560 1680 1800 1920 2040 2160 2280 2400
Random forest ROC
Random forest Precision

Como se puede observar, las métricas de los modelos van cambiando y evolucionando a medida que van llegando nuevos batches con nueva información, se puede observar que ciertas ocasiones las métricas de algunos modelos tienden a bajar, esto probablemente se deba a que cada batch puede poseer datos desbalanceados o que al unirse con con el resto de batches generen datos atípicos provocando los cambios en las precisiones de los modelos. Por otro lado, se puede considerar que algunos modelos alcanzaron a tener la misma precisión que los modelos del primer informe como se puede observar en las gráficas.

	Precisión	Área bajo el ROC
Regresión Logística	0.74	0.72
Árboles de Decisión	0.78	0.54
Random Forest	0.84	0.85

- **Cibergrafía**

    - **https://andres-plazas.medium.com/leer-y-escribir-datos-en-kafka-usando-python-2696154c3948**

    - **https://alvaromonsalve.com/2019/06/09/apache-kafka-apache-spark-un-ejemplo-de-spark-streaming-en-scala/**

    - **https://mtpradoc.medium.com/procesamiento-de-datos-en-streaming-usando-kafka-y-spark-structured-streaming-10f91b68b402**

    - **https://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html**

    - **http://kafka.apache.org/documentation.html#adminclientconfigs**