7 Proces de Streaming și Inferență ML în timp real cu PySpark

Ce am vrut să demonstrăm

În această ultimă secțiune, am implementat un **proces simplu de streaming** în PySpark, cu scopul de a simula un flux de date care vine în timp real și de a aplica pe fiecare batch un **model ML deja antrenat**. Concret, ne-am pus în pielea unei aplicații care monitorizează scorul de libertate al țărilor și îl actualizează automat pe măsură ce vin date noi.

Structura procesului

Componentă	Descriere
Sursă date	Fișiere CSV scrise treptat într-un folder (`stream_input`) din Google Drive
Model ML	`PipelineModel` salvat anterior (Logistic Regression)
Proces Streaming	Un `while` loop care verifică la fiecare 5 secunde dacă a apărut un fișier nou
Inferență	Modelul este aplicat imediat pe datele noi și returnează predicția `Low / Medium / High`

De ce e valoros

- **Simulare realistă** – fără `StreamingContext` , dar potrivită pentru Colab/local;
- **Aplicare practică** – putem înlocui oricând fișierele cu un stream Kafka real;
- **Util pentru scenarii reale** – predicții automate pentru organizații care vor să monitorizeze evoluția libertății în țări instabile.

```
!pip install -q pyspark


# ----------------- 1. IMPORTURI -----------------
import time
import os
import pandas as pd
from pyspark.sql import SparkSession
from pyspark.ml import PipelineModel
from pyspark.sql.functions import col
from pyspark.ml.feature import IndexToString


# ----------------- 2. START SPARK -----------------
spark = SparkSession.builder.appName("StreamingFreedomColab").getOrCreate()


# ----------------- 3. CĂI ABSOLUTE DIN GOOGLE DRIVE -----------------
model_path = "/content/drive/MyDrive/Master NLP/Anul 1 Semestrul 2/Big Data/Proiect Final/freedom_classifier_pipeline"
input_dir = "/content/drive/MyDrive/Master NLP/Anul 1 Semestrul 2/Big Data/Proiect Final/stream_input"



# ----------------- 4. ÎNCĂRCĂM MODELUL -----------------
print("Încărcăm modelul ML salvat...")
loaded_model = PipelineModel.load(model_path)
```

```
Încărcăm modelul ML salvat...
```

```
# ----------------- 5. LOOP DE STREAMING SIMULAT -----------------
print("Streaming pornit. Citim fișiere din:", input_dir)

processed_files = set()
start_time = time.time()
timeout = 60  # Rulăm streamingul timp de 60 secunde

while time.time() - start_time < timeout:
    files = [f for f in os.listdir(input_dir) if f.endswith(".csv") and f not in processed_files]

    for file in files:
        file_path = os.path.join(input_dir, file)
        print(f"\nFișier detectat: {file}")
        try:
            # 1. Citește fișierul și transformă-l în Spark DataFrame
            df = pd.read_csv(file_path).dropna()
            sdf = spark.createDataFrame(df)
            sdf = sdf.withColumn("year", col("year").cast("int"))  # conversie dacă e necesar

            # 2. Aplică modelul ML salvat
            predictions = loaded_model.transform(sdf)
```

```
            # 3. Decodează predicția în categorii text (Low, Medium, High)
            decoder = IndexToString(
                inputCol="prediction",
                outputCol="predicted_category",
                labels=["Low", "Medium", "High"]
            )
            final_df = decoder.transform(predictions)

            # 4. Afișează rezultatele
            results = final_df.select("countries", "region", "predicted_category").collect()
            for r in results:
                print(f"{r['countries']} ({r['region']}): {r['predicted_category']}")

            processed_files.add(file)

        except Exception as e:
            print(f"Eroare la {file}: {e}")

    time.sleep(5)

print("\nStreaming finalizat.")
```

```
Streaming pornit. Citim fișiere din: /content/drive/MyDrive/Master NLP/Anul 1 Semestrul 2/Big Data/Proiect Final/stream_input

Fișier detectat: input_auto_1.csv
Korea, Rep. (East Asia): Low
Moldova (Eastern Europe): Low
Morocco (Middle East & North Africa): Medium
Georgia (Caucasus & Central Asia): Low
Georgia (Caucasus & Central Asia): Low

Fișier detectat: input_auto_2.csv
Georgia (Caucasus & Central Asia): Low
Colombia (Latin America & the Caribbean): Low
Slovenia (Eastern Europe): Low
Slovenia (Eastern Europe): Low
Slovenia (Eastern Europe): Low

Fișier detectat: input_auto_3.csv
Ethiopia (Sub-Saharan Africa): Medium
Kazakhstan (Caucasus & Central Asia): Medium
Moldova (Eastern Europe): Low
Slovak Republic (Eastern Europe): Low
Jamaica (Latin America & the Caribbean): Low

Fișier detectat: input_auto_2 (1).csv
Georgia (Caucasus & Central Asia): Low
Colombia (Latin America & the Caribbean): Low
Slovenia (Eastern Europe): Low
Slovenia (Eastern Europe): Low
Slovenia (Eastern Europe): Low

Fișier detectat: input_auto_2 (2).csv
Georgia (Caucasus & Central Asia): Low
Colombia (Latin America & the Caribbean): Low
Slovenia (Eastern Europe): Low
Slovenia (Eastern Europe): Low
Slovenia (Eastern Europe): Low

Streaming finalizat.
```