**TFM – Flujo de datos (ALBERT 3 clases) y pipeline DVC**

**===========================================================**

**Autor: Pedro Martori**

**Proyecto: TFM-Sentidata**

**Descripción: Resumen auto-contenido del flujo de datos y stages DVC para la parte de modelado con ALBERT (3 clases).**

**-----------------------------------------------------------**

**ESQUEMA GENERAL (alto nivel)**

**-----------------------------------------------------------**

**Kaggle raw**

**└─► landing/raw ──► landing/delta ──► trusted\_clean**

**│**

**└─► (Spark/Delta cleaning + label3)**

**│**

**┌────────── train\_albert (frozen) ──────────┐**

**│ │**

**prepare\_test ──► test.parquet modelo/tokenizer**

**│ │**

**eval\_albert ────────────────────────────────────────┘**

**│**

**metrics, confusion, preds(test)**

**infer\_sephora (foreach particiones) ──► \*\_preds.parquet ──► merge ──► predictions\_all.parquet**

**│**

**metrics\_partitions.csv ◄──────────┘**

**metrics\_overall.json + confusion\_all.png**

**-----------------------------------------------------------**

**DETALLE DE STAGES DVC (Inputs → Proceso → Outputs)**

**-----------------------------------------------------------**

**1) descarga\_kaggle\_reviews [FROZEN]**

**• IN: scripts/data\_ingestion/descarga\_kaggle\_reviews.py**

**• PROCESO: descarga datasets RAW de Kaggle (Sephora/Ulta).**

**• OUT: data/landing/sephora/raw, data/landing/ulta/raw**

**• NOTA: stage congelada para no re-descargar.**

**2) conversion\_a\_delta [FROZEN]**

**• IN: data/landing/\*/raw**

**• PROCESO: conversión/normalización a Delta/Parquet.**

**• OUT: data/landing/sephora/delta, data/landing/ulta/delta**

**3) trusted\_cleaning**

**• IN: data/landing/\*/delta**

**• PROCESO (Spark + Delta):**

**– Filtra textos nulos o muy cortos.**

**– Convierte "rating" a entero y descarta nulos.**

**– Crea label3 (1–2→0 neg, 3→1 neu, 4–5→2 pos).**

**– Genera review\_id si no existe.**

**– Particiona por tamaño: reviews\_0\_250, …, reviews\_1250\_end.**

**• OUT: data/trusted/sephora\_clean/\* (+ ulta\_clean)**

**4) train\_albert\_subset\_0\_250 [FROZEN]**

**• IN: data/trusted/sephora\_clean/reviews\_0\_250, params.yaml, scripts/training/train\_albert.py**

**• PROCESO (HF Transformers + PyTorch):**

**– Split 85/15 estratificado por label3 (seed en params).**

**– Tokenización (max\_len), DataCollator, class weights.**

**– Entrenamiento (reanudable por checkpoint).**

**– Evaluación en validación y confusion.png (val).**

**• OUT (DVC-tracked dir): models/albert\_subset\_0\_250/**

**· model.safetensors, tokenizer.\*, metrics.json (val), confusion.png, checkpoints/\***

**• NOTA: al estar frozen, no re-entrena salvo que se fuerce.**

**5) prepare\_test\_albert\_subset\_0\_250**

**• IN: data/trusted/sephora\_clean/reviews\_0\_250**

**• PROCESO: genera TEST hold-out determinista (15%, --seed 42).**

**• OUT: reports/albert\_subset\_0\_250/test.parquet**

**• MOTIVO: eval rápida y reproducible sin Spark.**

**6) eval\_albert\_subset\_0\_250**

**• IN: models/albert\_subset\_0\_250, reports/albert\_subset\_0\_250/test.parquet**

**• PROCESO: carga el modelo entrenado, infiere en test y calcula métricas.**

**• OUT:**

**– reports/albert\_subset\_0\_250/metrics.json (ACC, F1-macro, F1 por clase)**

**– reports/albert\_subset\_0\_250/confusion.png (test)**

**– reports/albert\_subset\_0\_250/predictions.parquet (probs + preds)**

**7) infer\_sephora (foreach por particiones)**

**• IN: models/albert\_subset\_0\_250, data/trusted/sephora\_clean/{reviews\_\*}**

**• PROCESO: inferencia masiva batch en GPU/CPU sin reentrenar.**

**• OUT (uno por partición):**

**– reports/albert\_subset\_all/reviews\_\*\_preds.parquet**

**· columnas: id, pred\_index(0/1/2), pred\_label, p\_neg, p\_neu, p\_pos, true\_label (si rating/label3 existe)**

**8) merge\_infer\_sephora**

**• IN: todos los \*\_preds.parquet**

**• PROCESO: concatena todas las particiones.**

**• OUT: reports/albert\_subset\_all/predictions\_all.parquet**

**9) metrics\_partitions**

**• IN: reports/albert\_subset\_all/\*\_preds.parquet**

**• PROCESO: ACC y F1-macro por partición (si hay true\_label).**

**• OUT: reports/albert\_subset\_all/metrics\_partitions.csv**

**10) metrics\_overall**

**• IN: reports/albert\_subset\_all/predictions\_all.parquet**

**• PROCESO: métricas globales + matriz de confusión.**

**• OUT:**

**– reports/albert\_subset\_all/metrics\_overall.json**

**– reports/albert\_subset\_all/confusion\_all.png**

**-----------------------------------------------------------**

**DETALLES CLAVE (para la memoria)**

**-----------------------------------------------------------**

**• Etiquetado 3 clases (label3):**

**– rating 1–2 → 0 (neg), rating 3 → 1 (neu), rating 4–5 → 2 (pos).**

**– Consistente en cleaning, training, test e inferencias.**

**• Reproducibilidad:**

**– DVC versiona código + datos derivados.**

**– Stages de ingest y training congeladas (frozen).**

**– Test hold-out fijo con seed=42.**

**– Outputs separados: models/… (pesos) vs reports/… (preds/métricas).**

**• No fuga de datos:**

**– El test se define fuera del script de entrenamiento y queda fijado como Parquet.**

**• Generalización:**

**– Hold-out (eval\_albert\_\*): métricas y confusión en test.**

**– Intra-dominio (infer + metrics\_\*): rendimiento por particiones y global.**

**• Dispositivo (GPU/CPU):**

**– Scripts usan CUDA si está disponible; si no, CPU.**

**– Control rápido: CUDA\_VISIBLE\_DEVICES=0 (GPU) o vacío (CPU).**

**• Esquema típico de columnas en outputs de inferencia:**

**– id, pred\_index (0/1/2), pred\_label (si id2label), p\_neg, p\_neu, p\_pos, true\_label (opcional).**

**-----------------------------------------------------------**

**COMANDOS TÍPICOS (reproducción parcial o completa)**

**-----------------------------------------------------------**

**# (frozen) ingest + delta + clean + train**

**dvc repro train\_albert\_subset\_0\_250**

**# test fijo + evaluación hold-out**

**dvc repro prepare\_test\_albert\_subset\_0\_250**

**dvc repro eval\_albert\_subset\_0\_250**

**# inferencias masivas por particiones**

**dvc repro infer\_sephora**

**# o una sola partición:**

**dvc repro -f infer\_sephora@reviews\_0\_250**

**# merge + métricas**

**dvc repro merge\_infer\_sephora**

**dvc repro metrics\_partitions**

**dvc repro metrics\_overall**

**-----------------------------------------------------------**

**NOTAS FINALES**

**-----------------------------------------------------------**

**• El pipeline actual evita re-entrenar: el stage de training está frozen.**

**• Si cambias hyperparámetros o scripts clave, DVC detecta diferencias y reproduce lo necesario.**

**• Los archivos grandes (modelos, preds) se versionan con DVC (no con Git).**

**• Para la entrega del TFM, incluye: metrics.json y confusion.png (test), metrics\_partitions.csv, metrics\_overall.json, confusion\_all.png, predictions\_all.parquet, y el dvc.yaml/dvc.lock.**

**ESQUEMA DE ARVHIVOS**

**(venv) pedro@PVS:~/MASTER\_BIGDATA/TFM/TFM-Sentidata$ tree -L 4**

**.**

**├── README.md**

**├── data**

**│ ├── exploitation**

**│ │ ├── analisis\_final**

**│ │ ├── dashboards\_data**

**│ │ ├── modelos\_input**

**│ │ └── products**

**│ ├── landing**

**│ │ ├── sephora**

**│ │ │ ├── delta**

**│ │ │ └── raw**

**│ │ └── ulta**

**│ │ ├── delta**

**│ │ └── raw**

**│ └── trusted**

**│ ├── sephora\_clean**

**│ │ ├── product\_info**

**│ │ ├── reviews\_0\_250**

**│ │ ├── reviews\_1250\_end**

**│ │ ├── reviews\_250\_500**

**│ │ ├── reviews\_500\_750**

**│ │ └── reviews\_750\_1250**

**│ └── ulta\_clean**

**│ └── Ulta Skincare Reviews**

**├── dvc.lock**

**├── dvc.yaml**

**├── mlruns**

**│ └── 0**

**│ ├── 8854a58292f04b77bef57111634c51b0**

**│ │ ├── artifacts**

**│ │ ├── meta.yaml**

**│ │ ├── metrics**

**│ │ ├── params**

**│ │ └── tags**

**│ └── meta.yaml**

**├── models**

**│ ├── albert\_subset**

**│ ├── albert\_subset\_0\_250**

**│ │ ├── checkpoint-11967**

**│ │ │ ├── config.json**

**│ │ │ ├── model.safetensors**

**│ │ │ ├── optimizer.pt**

**│ │ │ ├── rng\_state.pth**

**│ │ │ ├── scheduler.pt**

**│ │ │ ├── special\_tokens\_map.json**

**│ │ │ ├── spiece.model**

**│ │ │ ├── tokenizer.json**

**│ │ │ ├── tokenizer\_config.json**

**│ │ │ ├── trainer\_state.json**

**│ │ │ └── training\_args.bin**

**│ │ ├── checkpoint-6000**

**│ │ │ ├── config.json**

**│ │ │ ├── model.safetensors**

**│ │ │ ├── optimizer.pt**

**│ │ │ ├── rng\_state.pth**

**│ │ │ ├── scheduler.pt**

**│ │ │ ├── special\_tokens\_map.json**

**│ │ │ ├── spiece.model**

**│ │ │ ├── tokenizer.json**

**│ │ │ ├── tokenizer\_config.json**

**│ │ │ ├── trainer\_state.json**

**│ │ │ └── training\_args.bin**

**│ │ ├── config.json**

**│ │ ├── confusion.png**

**│ │ ├── metrics.json**

**│ │ ├── model.safetensors**

**│ │ ├── special\_tokens\_map.json**

**│ │ ├── spiece.model**

**│ │ ├── tokenizer.json**

**│ │ ├── tokenizer\_config.json**

**│ │ └── training\_args.bin**

**│ ├── mlflow\_runs**

**│ │ └── 0**

**│ │ └── meta.yaml**

**│ └── model\_lr\_price.pkl**

**├── notebooks**

**├── params.yaml**

**├── reports**

**│ ├── albert\_subset\_0\_250**

**│ │ ├── confusion.png**

**│ │ ├── metrics.json**

**│ │ ├── predictions.parquet**

**│ │ └── test.parquet**

**│ ├── albert\_subset\_all**

**│ │ ├── confusion\_all.png**

**│ │ ├── metrics\_overall.json**

**│ │ ├── metrics\_partitions.csv**

**│ │ ├── predictions\_all.parquet**

**│ │ ├── reviews\_0\_250\_preds.parquet**

**│ │ ├── reviews\_1250\_end\_preds.parquet**

**│ │ ├── reviews\_250\_500\_preds.parquet**

**│ │ ├── reviews\_500\_750\_preds.parquet**

**│ │ └── reviews\_750\_1250\_preds.parquet**

**│ └── figures**

**├── requirements.txt**

**├── scripts**

**│ ├── data\_cleaning**

**│ │ ├── conversion\_a\_delta.py**

**│ │ ├── trusted\_clean\_driver.py**

**│ │ └── trusted\_clean\_single.py**

**│ ├── data\_ingestion**

**│ │ └── descarga\_kaggle\_reviews.py**

**│ ├── data\_integration**

**│ ├── evaluation**

**│ │ ├── merge\_parquets.py**

**│ │ ├── metrics\_overall.py**

**│ │ └── metrics\_partitions.py**

**│ ├── main.py**

**│ ├── setup**

**│ │ ├── dvc\_run.sh**

**│ │ └── verificar\_dvc.sh**

**│ ├── training**

**│ │ ├── check\_parquet\_columns2.py**

**│ │ ├── conteo.py**

**│ │ ├── evaluate\_albert.py**

**│ │ ├── infer\_albert\_all.py**

**│ │ ├── prepare\_test\_parquet.py**

**│ │ └── train\_albert.py**

**│ └── visualization**

**├── tests**

**└── tmp\_trainer**

**Train / Val / Test (partición reviews\_0\_250)**

**Total en reviews\_0\_250 (confirmado por logs): 600.648 reviews**

**(apareció como Map: … 600648/600648)**

**Split de entrenamiento (85/15):**

**Train: 510.550**

**Val: 90.098**

**Test (hold-out) fijo (15%): 90.098**

**(apareció como test.parquet (filas=90098))**

**Nota: el train/val es el split interno del script de training; el test viene de prepare\_test\_\* y tiene el mismo tamaño (15%) por diseño.**

**TOTAL inferencia: 1,092,201:**

1. **reviews\_0\_250: 600,648**
2. **reviews\_250\_500: 206,411**
3. **reviews\_500\_750: 116,025**
4. **reviews\_750\_1250: 119,206**
5. **reviews\_1250\_end: 49,911**

**Filas: 1092201 IDs únicos: 1092196**

**training**

Gráfico, Gráfico de barras

El contenido generado por IA puede ser incorrecto.

**Test**

**Gráfico, Gráfico de rectángulos

El contenido generado por IA puede ser incorrecto.**

**Test normalized**

**Gráfico, Gráfico de rectángulos

El contenido generado por IA puede ser incorrecto.**

== reports/albert\_subset\_0\_250/metrics.json ==

metric value

accuracy 0.8976114897112033

precision\_macro 0.7528442052424703

recall\_macro 0.8560101087672223

f1\_macro 0.7833942978912433

f1\_negative 0.8559661977857411

f1\_neutral 0.5428600971978079

f1\_positive 0.9513565986901806

Gráfico, Gráfico de rectángulos

El contenido generado por IA puede ser incorrecto.

Gráfico, Gráfico de rectángulos

El contenido generado por IA puede ser incorrecto.

== reports/albert\_subset\_all/metrics\_overall.json ==

metric value

n 1092201

accuracy 0.8903562622630816

f1\_macro 0.7698516509004888