Memoria del Proyecto — Predicción de Rotación de Empleados

Autora: Beatriz Velayos · **Bootcamp:** Big Data · AI · Machine Learning · **Tag:** v1.0-entrega

1) Idea y utilidad para la empresa

Predecir qué empleados tienen mayor probabilidad de causar baja (rotación voluntaria) para priorizar acciones de retención (revisiones salariales, planes de carrera, conciliación). Salidas operativas:

- Lista priorizada de empleados por probabilidad (con umbral operativo).
- Dashboard de negocio (Power BI) con KPIs y drivers interpretables.

2) Objetivos académicos (Bootcamp)

- Pipeline reproducible (Docker) de calidad → ETL → ML → métricas.
- Entrenar y comparar modelos baseline (LogReg, Random Forest).
- Persistir resultados y exponerlos en un dashboard listo para negocio.
- Entregar código versionado, notebooks HTML, PBIX y esta memoria.

3) Datos utilizados

- **Histórico HR**: WA_Fn-UseC_-HR-Employee-Attrition.csv (1470 empleados; objetivo `Attrition` en {Yes, No}).
- **Encuesta de clima**: encuesta_clima.csv (Engagement, Satisfaction, WorkLifeBalanceSurvey, ManagerRelationship, RemoteWorkSatisfaction).
- **Clave de unión**: `EmployeeNumber`.

Variables derivadas: `overtime_flag`, `income_yearly = MonthlyIncome * 12`, `tenure_ratio = YearsAtCompany / TotalWorkingYears`.

Clase positiva: 237 “Yes” (~16,1%).

4) Arquitectura y estructura

- Servicios: Jupyter (ML), Spark (calidad/ETL), Postgres (persistencia opcional).
- Estructura clave: `bi/` (PBIX), `docs/` (HTML + PDF), `scripts/` (pipeline), `data/`, `output/` (modelos y métricas).

5) Metodología y pasos

5.1 Preparación

Contenedores con `docker compose up -d` y trabajo en JupyterLab.

5.2 Calidad de datos (Spark)

Script: `scripts/check_data_quality.py`. Comprobaciones: nulos en clave, duplicados, fracción de nulos por columna (< 0,25), dominio del objetivo. Resultado: **PASS**.

5.3 ETL (Spark)

Script: `scripts/etl_attrition.py`. Join por `EmployeeNumber`, tipificación y normalización ligera, creación de features y salida a `data/processed/employee_attrition.parquet` (1470 filas; 16,1% positiva).

5.4 Preprocesado

Script: `scripts/preprocess.py`. Numéricas (StandardScaler) + categóricas (One-Hot). Se guarda el *transformer* en `output/models/transformer.joblib`.

5.5 Entrenamiento y comparación

Script: `scripts/train_ml.py`. Modelos: Logistic Regression (interpretable) y Random Forest (no lineal).

- **LogReg**: ROC-AUC ≈ 0,817 · PR-AUC ≈ 0,563 · F1* ≈ 0,529 · thr* ≈ 0,732.
- **RF**: ROC-AUC ≈ 0,805 · PR-AUC ≈ 0,536 · F1* ≈ 0,535 · thr* ≈ 0,22.

Elección: **LogReg** por mejor PR-AUC e interpretabilidad. Métricas agregadas en `output/metrics/model_compare.json`.

5.6 Persistencia

Opción Postgres (tabla `predictions_latest`) o CSV para Power BI (`output/bi/feature_effects.csv`) si se evita conexión directa.

5.7 Dashboard (Power BI)

- Relación activa por `EmployeeNumber` entre predicciones y atributos.
- Tabla Dim/Dum Modelo (desconectada) para Slicer de `model_name` (selección única).

Páginas: (1) Resumen & KPIs; (2) Drivers (importancias/efectos); (3) Departamentos; (4) Puestos.

6) Resultados

- Tasa histórica de rotación: ~16,1%.
- Modelo principal (LogReg): ROC-AUC ≈ 0,817 · PR-AUC ≈ 0,563 · F1* ≈ 0,529.

- Drivers típicos de riesgo: `OverTime` alto, `YearsAtCompany` bajos, `MonthlyIncome` bajo, percepciones de conciliación/satisfacción peores.

7) Entregables

- Repositorio Git con código, modelos y métricas (`tag v1.0-entrega`).
- Notebooks en HTML (`docs/01_EDA_Attrition.html`, `02_Modelado_Baseline.html`, `05_Dashboard_KPIs.html`).
- Dashboard Power BI (`bi/Employee_Attrition_Dashboard.pbix`).
- Memoria del proyecto (este PDF) en `docs/`.

8) Limitaciones y mejoras

- Clase minoritaria: calibración de probabilidades y umbral por costes.
- Más datos (absentismo, desempeño) y validación más robusta.
- MLOps: CI/CD, model registry, monitorización de *drift*.

9) Guía rápida

```
docker compose up -d
docker compose run --rm jupyter python /scripts/check_data_quality.py \
  --input1 /data/raw/WA_Fn-UseC_-HR-Employee-Attrition.csv \
  --input2 /data/raw/encuesta_clima.csv --key EmployeeNumber \
  --max-null-frac 0.25 --spark-master local[*]

docker compose run --rm jupyter python /scripts/etl_attrition.py \
  --input1 /data/raw/WA_Fn-UseC_-HR-Employee-Attrition.csv \
  --input2 /data/raw/encuesta_clima.csv --key EmployeeNumber \
  --outdir /data/processed/employee_attrition.parquet --spark-master local[*]

docker compose run --rm jupyter python /scripts/preprocess.py \
  --input /data/processed/employee_attrition.parquet \
  --out /output/models/transformer.joblib

docker compose run --rm jupyter python /scripts/train_ml.py \
  --input /data/processed/employee_attrition.parquet --model logreg
docker compose run --rm jupyter python /scripts/train_ml.py \
  --input /data/processed/employee_attrition.parquet --model rf
```

10) Conclusión

Se entrega un sistema reproducible para predecir rotación, con métricas transparentes y dashboard para priorizar acciones de retención: a quién ayudar primero y por qué, con trazabilidad técnica para evolucionarlo.