Rev 0.2
August 7th 2025

Data Scientist – Take-Home Exercise 1

1 Overview

1.1 Your Mission

Step into the role of a Staff-level Data Scientist at **Acme Streaming**. Your charter is to turn raw behavioural logs into trusted insight, a predictive service, and an executive-calibre narrative that shapes product and marketing strategy. We have scoped the work to fit within **6–8 focused hours**—ample room to demonstrate judgement without monopolising your weekend.

Your task: Demonstrate, using any combination of Spark, advanced modelling, visual analytics, and statistical reasoning, that you can isolate, quantify, and convincingly defend the primary drivers of early subscriber churn in the dataset.

AI-based development is encouraged: This task enforces responsible LLM usage, leveraging productivity without compromising security, reliability, or quality. In other words, this is very hard to do in all its parts without assisted help within the indicated timeframes. This is done on purpose: in Yard Reaas we encourage all cost-effective AI accelerations for doing your work! Even those you do not build yourself (though we prefer yours indeed)!

We included some tips on how to use LLMs in the last section of this document

1.2 Business Problem & Goals

Recent signals suggest **early churn**—customers who cancel within their first 90 days—has quietly crept above forecast. Leadership needs a prototype that:

1. **Measures** churn magnitude and seasonality.
2. **Predicts** which active subscribers are at risk in the coming month.
3. **Explains** the "why" in language that non-technical stakeholders understand.

Your workflow should connect the dots end-to-end: **ingest → explore → model → explain → recommend**.

1.3 Tools & Spark Setup

You may use **Python, R, or Scala**. All tooling must be open-source and runnable on a laptop. The processing backbone is **Apache Spark 3.4+**; a local session is perfectly acceptable. If you prefer a managed environment, any of the no-cost options below will suffice.

Zero-Cost Spark Options

1 AWS EMR Serverless (Free Tier) • Console → **EMR Serverless** → *Create application* (Spark 3.4). • Set **Max capacity** to **1 driver + 2 executors @ 1 vCPU / 4 GiB RAM**—well inside the 50 vCPU-hr &

10 GB-hr free allowance. • Upload ≤ 1 GB sample data & notebooks to an S3 bucket (also free tier). • Submit via `spark-submit` or EMR Studio, then **stop the app** when finished.

2 Databricks Community Edition • Sign up at databricks.com (free). • Launch a **single-node cluster** (15 GB RAM, 2 vCPU). • Attach a notebook and install extra libraries with `%pip install`.

2 Data

Dataset: **Wikimedia Pageviews – January 2025** (≈ 70GB GB compressed) — https://dumps.wikimedia.org/other/pageviews/2025/2025-01/

3 Assignment

3.1 Data Engineering (Spark)

- Ingest raw files in an optimised ingestion pipeline and materialise a **bronze → silver → gold** layer in Parquet or Delta.
- Document schema design, partitioning, and data-quality guards (late records, outliers, etc.).

3.2 Exploratory Analysis

- Compute engagement KPIs (DAU/WAU/MAU, session length, content diversity, etc.).
- Visualise trends to inform feature ideas and retention hypotheses.
- Highlight at least **three data quirks** (seasonality, sparsity, leakage) and how you mitigated them.

3.3 Feature Engineering & Modelling

- Cast churn as a binary classification problem (define your churn window clearly).
- Train one selected algorithm of your choice (Warning: you will need to explain why!).
- Run systematic hyper-parameter search (Cross-Validation, Hyperopt, or Bayesian) and compare ROC-AUC / PR-AUC.
- Use model-agnostic explainers (SHAP, permutation importance) to surface top predictive drivers.

3.4 BI Dashboard ⇄ Data Story

- Build an **interactive dashboard** in **Superset, Metabase, or Redash** that *is itself* the narrative for your presentation. At minimum include:

  1. Overall churn and retention funnel.
  2. Cohort heat-map (join month × survival).
  3. Slice-and-dice filters for tenure, engagement, geography, etc.
  4. Explainable model outputs (risk scores + key drivers visualised). * **NO SLIDES REQUIRED**, the dashboard **IS** the narrative!

4 Deliverables

You must provide the following artifacts:

1. **Git repository** with code, notebooks, and `ENVIRONMENT.md`.

2. **Interactive BI dashboard link** (or static screenshots if self-hosting isn't feasible).
3. **Analytical results document** — a concise, scientific-style essay (PDF) detailing methodology, experiments, and findings.

5 Evaluation Matrix

Area	Weight	What We Look For
Spark Engineering	20%	Robust pipelines, performance-aware design
Statistical Modelling & Tuning	30%	Sound baselines, feature craftsmanship, reproducible experiments
BI Dashboard & Narrative	30%	Clear, interactive storytelling through the dashboard
Analytical Essay	10%	Rigour, clarity, and critical discussion of results & limitations
Code Quality & Dev-Ops	10%	Readability, modularity, tests, turnkey setup

6 Notes on LLM Usage

- **Workflow acceleration with LLM-based coding is strongly suggested** for boilerplate generation, tests suggestions, or prototyping complex logic and presentations.
- **Query any LLM in English to activate the best reasoning layer…**, where model performance is strongest: models are heavily trained on English data, and studies confirm English prompts significantly outperform others in accuracy and reliability (arxiv.org, openreview.net).
- **…or query them in any language but ask them to reason in English!** A pro-trick to activate their best reasoning layer in another language: ask them to translate your input in English first, reason in English, then draft an ouput in English and finally translate it back the input language!
- **Verify code info, dependencies and imports** before using them. Almost 20% of LLM-generated code references non-existent packages, creating supply-chain risks (e.g., "slopsquatting") (infosecurity-magazine.com, techradar.com).
- **Test generated code immediately** to catch syntax errors from hallucinated methods; logical edge cases often require deeper review.
- **Refine iteratively**: use prompts to enhance performance, edge-case handling, and clarity—review each output manually.
- **Annotate LLM-generated sections** clearly in comments or commit history to aid peer review and ensure transparency.
- **Apply standard engineering safeguards**: run static analysis, dependency scans, peer code reviews, and CI tests —LLM-generated code should not bypass these.