# Uplift Engine 2.1 — Project Summary

# Uplift Engine 2.1 — Project Summary

Author: Team Uplift

Date: 2025-10-19

Executive summary

-----------------

This document summarizes the Uplift Engine 2.1—an end-to-end, production-ready data platform designed to optimize promotion campaigns for VPBank through causal AI and MLOps practices. The summary is written at a master's-level technical depth and is intended for cross-functional audiences: product owners, cloud engineers, and ML scientists.

1. Business context & goals

---------------------------

* Problem statement: Traditional propensity models yield low ROI because they do not measure the causal impact of interventions. Marketing budgets are wasted on "Sure Things" and "Lost Causes." The bank needs a system that identifies "Persuadables"—customers who will convert because of the promotion.
* Business objectives:
* - Maximize incremental conversion and profit per campaign.
* - Reduce wasted marketing spend by at least X% (pilot target: 20%).
* - Provide auditable treatment assignment and measurement for regulatory and business oversight.

2. Product goals

----------------

* Deliver a minimal viable product (MVP) within 4-6 weeks: data ingestion, feature engineering, uplift model training, API for scoring, and basic ROI/Qini dashboards.
* Deliver a production pilot in 3 months: hardened MLOps, Feature Store, automated retraining, monitoring and guardrails.

3. Key concepts and algorithms

------------------------------

* Uplift Modeling (Causal AI): focused on estimating treatment effect at individual level. Preferred algorithms:
* - UpliftRandomForestClassifier (causalml) — fast baseline and interpretable.
* - Double Robust Learner (DR-Learner) — reduces bias and variance with orthogonalization.
* - CatBoost Uplift — robust to categorical features and often performant on tabular data.
* - EconML (for advanced causal inference and heterogeneous treatment effect estimation).
* Knapsack Optimization: to select treatment population under budget constraints, maximizing expected incremental profit rather than naive top-K.
* Guardrails: confidence-interval-based thresholds, DNC lists, and business rules to ensure Do-No-Harm.

4. Architecture overview

------------------------

* Data Lake (S3): central raw and processed storage.
* Ingestion: API, batch ETL, Kinesis for near-real-time events.
* Feature Store: SageMaker Feature Store (or open-source alternative) for training-serving consistency.
* Training: SageMaker or local notebooks for prototyping; EMR Serverless for heavy Spark workloads.
* Orchestration: Step Functions for managing complex pipelines (optional).
* Serving: API Gateway + Lambda (Provisioned Concurrency) for real-time scoring; batch scoring via EMR/Glue for large-scale campaigns.
* Monitoring & Dashboarding: QuickSight (or Grafana + Prometheus) for ROI, Qini, and model performance.

5. Why these choices (trade-offs)

--------------------------------

* S3 + Feature Store: decouples storage and feature serving. Ensures training-serving parity and reproducibility.
* Lambda + Provisioned Concurrency: keeps the serving path serverless and low-latency; PC eliminates cold starts at the cost of reserved concurrency fees.
* Tiered batch (Glue / EMR Serverless): Glue for routine jobs (cost-effective), EMR Serverless for heavy feature engineering jobs (performance), balancing cost and performance.
* Causal algorithms: uplift models directly optimize incremental effect; DR-Learner and EconML reduce confounding bias and provide better uncertainty estimates, important for business trust.

6. Process & MLOps

-------------------

* Data validation & lineage: incoming events are validated and catalogued; Feature Store records feature versions.
* CI/CD for models: use GitHub Actions or CodePipeline to run unit tests, model training smoke tests, and package Lambda artifacts.
* Model registry & artifacts: version models in S3 or SageMaker Model Registry; keep metadata (training data hash, features used).
* Monitoring: data drift alerts, model performance degradation, campaign-level ROI tracking.
* Retraining policy: scheduled retraining weekly or trigger-based when drift exceeds threshold.

7. Security & compliance

------------------------

* Use IAM roles with least privilege; never check secrets into repo.
* Encrypt data at rest (S3 SSE) and in transit (HTTPS).
* Apply data retention and anonymization policies where required by regulation.

8. Risks and mitigations

------------------------

* Risk: Cold-start / latency affecting SLA. Mitigation: Provisioned Concurrency for Lambda; compute critical features in app layer.
* Risk: Glue job OOM/timeouts. Mitigation: tiered processing with EMR Serverless for heavy jobs.
* Risk: Model harming customer segments. Mitigation: Do-No-Harm guardrails, CI-based tests, and manual review for VIP segments.

9. Roadmap & milestones

-----------------------

* Week 0-2: Data integration & sample dataset; basic uplift baseline model.
* Week 3-4: Real-time scoring API + basic dashboard; MVP demo.
* Month 2: Feature Store + automated retraining; A/B pilot.
* Month 3: Production pilot with PC-enabled Lambda, tiered batch, and monitoring.

10. Deliverables for handoff

---------------------------

* `docs/VPBank\_UpliftEngine\_Guide.md` — full technical guide.
* `docs/presenter\_notes.md` — slides + speaker notes.
* `docs/architecture\_notes.md` — instructions for updating architecture diagram.
* `docs/UpliftEngine\_Presentation\_Summary.md` and (pending) `docs/UpliftEngine\_Presentation\_Summary.docx` — the Word summary.

Appendix A — Suggested slide structure for team presentation

* Title
* Problem & Context
* Architecture (diagram)
* Method & Models
* Demo (Qini / API call)
* Business Impact & ROI
* Roadmap & Ask

To-do for finishing the Word document

------------------------------------

1. Internal review & refine the markdown draft (this file). (assigned: Team Lead)

2. Add one-page case study (pilot numbers) with simulated ROI table. (assigned: ML Scientist)

3. Finalize language and citations (assigned: Team Lead / ML Scientist)

4. Convert to .docx and add TOC, headings, and company cover page. (assigned: Cloud Engineer)

5. Final review & sign-off. (assigned: Team Lead)