**Project Report Summary**

**Purpose & Outcome**

**Purpose:** Build a reproducible forecasting pipeline that quantifies how Steam discount events and major content updates drive one-month changes in concurrent player counts across 50 popular titles. The goal is to empower developers with data-driven insights on optimal sale timing and update scheduling.

**Outcome:** A tuned Random Forest model that explains ~30 % of the variance in month-over-month player-count changes (Hold-out R² ≈ 0.30, MAE ≈ 3 037, RMSE ≈ 6 790), packaged in a self-contained pipeline for generating actionable, month-ahead forecasts.

**1. Data Collection & Features**

* **Sources:**
  + **SteamCharts** for historical avg\_players
  + **Steam Store API** for price and genres
  + **Steam News API** for had\_update flags
* **Engineered Features:**
  + **Lag/Trend:**
    - lag1: previous month’s players
    - ma3: 3-month moving average
    - std3: 3-month rolling standard deviation
  + **Business Flags:**
    - sale\_flag: binary for January, February, July (historical high-gain months)
    - had\_update: binary for major patches/DLCs
  + **Pricing:** price\_delta (month-over-month price change)
  + **Calendar:** month\_num (1–12)

**2. Modeling & Tuning**

* **Models Compared:** Random Forest, Decision Tree, kNN
* **Best Performer:** Random Forest Regressor
* **Hyperparameters (via time-series CV):**
  + n\_estimators=300
  + max\_depth=5
  + min\_samples\_leaf=3
  + max\_features='sqrt'
* **Hold-out Results (≥ Jul 2024):**
  + **R²:** 0.295
  + **MAE:** 3 037 players
  + **RMSE:** 6 790 players

**3. Key Findings**

* **Primary Drivers:**
  + ma3, lag1, and std3 together account for ~90 % of feature importance.
  + sale\_flag and price\_delta contribute negligibly, suggesting fixed-month flags and raw price changes are too coarse.
* **Patterns:**
  + The model reliably captures broad “ups and downs” for volatile titles (e.g. Stardew Valley).
  + Seasonality encoded simply as month\_num adds some signal (≈ 5 %), but richer cyclic encodings could improve performance.

**4. Deployment**

* **Artifacts Saved:**
  + rf\_player\_change.pkl (model)
  + scaler\_player\_change.pkl
* **Pipeline:**
  + export\_data.py → builds full feature CSV
  + train\_model.py (notebook) → trains & evaluates Random Forest
  + predict\_change.py → outputs predictions.csv for the upcoming month

**5. Next Steps**

* **Refine Sale Flags:** scrape actual discount percentages rather than fixed months.
* **Engineer Richer Calendar Features:** sine/cosine cycles or one-hot month dummies.
* **Explore Multi-month Targets:** smoothing the target (e.g. 3-month change) to reduce noise.
* **Advanced Time-Series Models:** ARIMA, Prophet, or causal impact to model duration of sale/update effects.