Income Prediction & Segmentation Project Report

Nainika Saha
Client: Walmart Retail Analytics
Data: CPS (U.S. Census Bureau), 1994–1995

1) Objectives

1. Build a classifier that predicts whether an individual earns > $50,000 based on ~40 demographic and employment features.
2. Build a segmentation model to group individuals into distinct, marketing-relevant clusters, and explain how these groups differ and can be actioned.

I aimed for a pipeline that is simple, reproducible, and business-friendly: minimal assumptions, strong baselines, weighted evaluation, and clear explanations of trade-offs.

2) Data Understanding & Exploration

Source & structure. The dataset contains weighted records from the CPS (1994–95) with 40 features, a weight column (survey sampling weights), and a label indicating whether income is above or below $50K. Features span:

- **Demographics:** age, sex, race, marital status, education.
- **Employment:** class of worker, detailed industry/occupation codes, union, weeks worked, full/part-time status.
- **Household context:** family composition, children in household.
- **Migration/residence:** region, state, sunbelt, move indicators.
- **Other:** capital gains/losses, dividends, citizenship, veteran's benefits.

Label quirks. The raw label in this extract appears as '- 50000.' and '50000+.' (with spaces/punctuation). I normalized to canonical <=50K / >50K. This is implemented in src/data_utils.py.

"Not in universe." CPS responses use "Not in universe" when a question doesn't apply (e.g., occupation for children, class-of-worker for retirees). This is not missing data; it's informative. Segments dominated by "Not in universe" values represent **non-labor-force** groups (children, retirees, homemakers).

Class balance. After applying weights, **~89%** of the population is <=50K and **~11%** is >50K. This matters for evaluation (accuracy alone is misleading; I focus on ROC-AUC and PR-AUC and report class-specific precision/recall).

Basic checks:

- Trimmed/trailing spaces present in some categories; I stripped whitespace across string columns.
- Numeric fields (e.g., weeks worked in year) are preserved and later standardized.
- The year feature primarily takes 1994/1995 values (as expected).
- No label leakage features (e.g., a direct wage value) are included in the model; wage per hour exists but is often zero for non-workers and is not a direct target proxy.

3) Preprocessing Approach

Goals: keep it transparent and generalizable; avoid heavy feature engineering that hides assumptions.

Steps implemented (see src/preprocess.py and src/data_utils.py):

- **Column names** loaded from census-bureau.columns.
- **String cleanup**: strip whitespace; normalize label strings to <=50K / >50K.
- **Type handling**:
    - Categorical → **OneHotEncoder** with handle_unknown="ignore".
    - Numeric → **StandardScaler** (mean 0, unit variance).
- **Weights**: I use the CPS weight column as sample_weight during training and evaluation to ensure metrics and learned decision boundaries reflect **population**, not the sampling scheme.
- **Split**: **Stratified 80/20** train/test split (random_state=42) to preserve label ratios and keep results reproducible.

Why not heavy feature engineering? The objective is to show thought process and deliver a robust baseline. One-hot + scaling is a clean, defensible default that works across both the classifier and the clustering pipeline. If I needed to push for SOTA metrics, I'd add target encoding (with CV), interaction features, or domain-guided binning—but those introduce extra assumptions and review overhead.

4) Model Architecture & Training

4.1 Classifier

Models trained:

- **Logistic Regression (SAGA)** with L2 (baseline; interpretable).
- **LightGBM (gradient boosted trees)** (final; strong on mixed/tabular data).

Why these two?

- Logistic regression sets a simple, transparent baseline and checks that data preprocessing behaves.
- LightGBM handles heterogeneous features, non-linearities, and sparse one-hot representations very well, usually a strong choice for tabular datasets with many categorical levels.

Training details:

- **Weights**: CPS weight passed to .fit(...) and to all metric functions.
- **Imbalance**: I did not oversample or undersample. I want population-representative probabilities. LightGBM naturally tolerates skew. If recall must be increased later, I would adjust the probability threshold or consider class-cost parameters with post-hoc calibration.

Evaluation protocol:

- Holdout test set (20%), stratified.
- Metrics:
    - **Accuracy** (for context), **ROC-AUC** (rank quality),
    - **PR-AUC** (class-imbalance sensitive),
    - **Precision/Recall/F1** for the positive class,
    - **Weighted confusion matrix**.
- Reporting: results/classifier_metrics.json and simple permutation-based block importances (coarse, mainly for sanity).

Classifier results (LightGBM, test, weighted):

- **Accuracy:** 95.72%
- **ROC-AUC:** 0.955
- **PR-AUC:** 0.699
- **Precision (positive):** 0.759
- **Recall (positive):** 0.493
- **F1:** 0.598
- **Confusion matrix (weighted counts)**
    - True ≤50K predicted ≤50K: **64,250,904.6**
    - True ≤50K predicted >50K: **702,197.0**
    - True >50K predicted ≤50K: **2,273,103.8**
    - True >50K predicted >50K: **2,210,376.8**

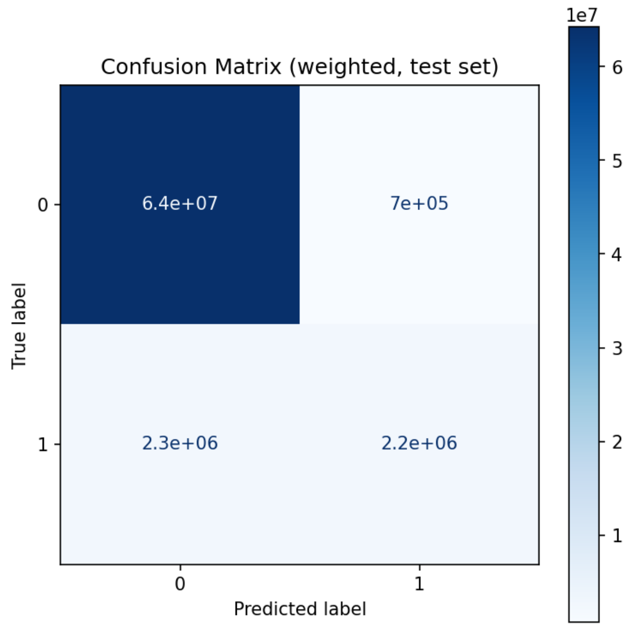

Figure 1. Weighted confusion matrix showing classification performance. High accuracy overall, with lower recall on the >50K group.

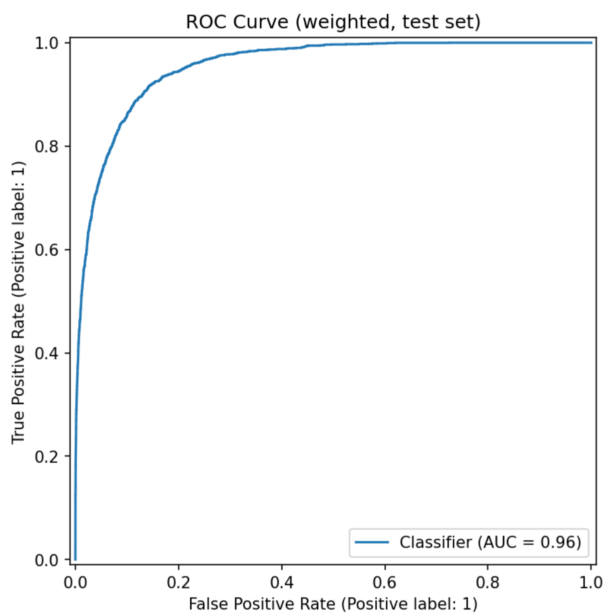

Figure 2. ROC curve with AUC = 0.955, demonstrating strong separation between income groups.

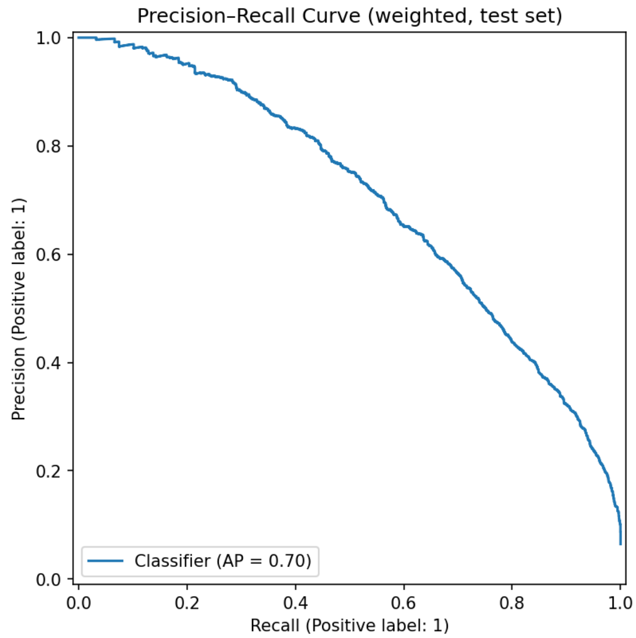

Figure 3. Precision–Recall curve highlighting the precision/recall trade-off under class imbalance (11% >50K share).

Interpretation and business judgment:

- The model is conservative on the scarce class: when it predicts high-income, it's usually right (precision ~0.76), but it misses some high-income individuals (recall ~0.49). This is expected with ~11% positives.
- I chose not to artificially balance the classes because Walmart needs population-realistic outputs (for campaign sizing and ROI projections).
- If the business goal shifts to maximizing recall (e.g., "don't miss high-income folks"), I'd:
    1. Tune the probability threshold to trade precision for recall;
    2. Optionally set LightGBM's class-weighting (e.g., is_unbalance=True or scale_pos_weight) and then calibrate probabilities;
    3. Evaluate business impact on CAC/CPA and LTV.

Quick feature sanity (coarse permutation):

- As expected, weeks worked, education level, full-time/part-time, and professional/medical occupations light up. Capital gains/dividends are also strong signals when present. This aligns with the segmentation's high-income cluster.

4.2 Segmentation

Goal: provide actionable groups for marketing.
Approach: K-Means on the same preprocessed features (label/weight dropped), k ∈ [3..8], pick k by **silhouette** on a random sample, then refit on the full transformed data.

Why K-Means? It's transparent, fast, and easy to explain to non-technical stakeholders. Silhouette ≈ 0.2 is common for socio-demographic data with many categorical OHE features; I'm using clusters descriptively to support targeting, not as hard silos for years.

Selection outcome: Best k = 8, **silhouette ≈ 0.220**.

Segment sizes (row counts):
Seg0: 23,605 | Seg1: 23,435 | Seg2: 47,921 | Seg3: 26,082
Seg4: 26,558 | Seg5: 3,793 | Seg6: 392 | Seg7: 47,737

Income share by segment (share_>50K):

- Seg0: **0.02** (retiree/homemaker profile)
- Seg1: **0.02** (similar retiree/homemaker profile)
- Seg2: **0.11** (retail/clerical, HS grads, married males)
- Seg3: **0.00** (children)
- Seg4: **0.00** (children, nonmovers)
- Seg5: **0.31** (private/professional HS grads; mid-income)
- Seg6: **0.88** (bachelor's+, medical/professional; small but affluent)
- Seg7: **0.10** (retail/clerical; large, value-driven)

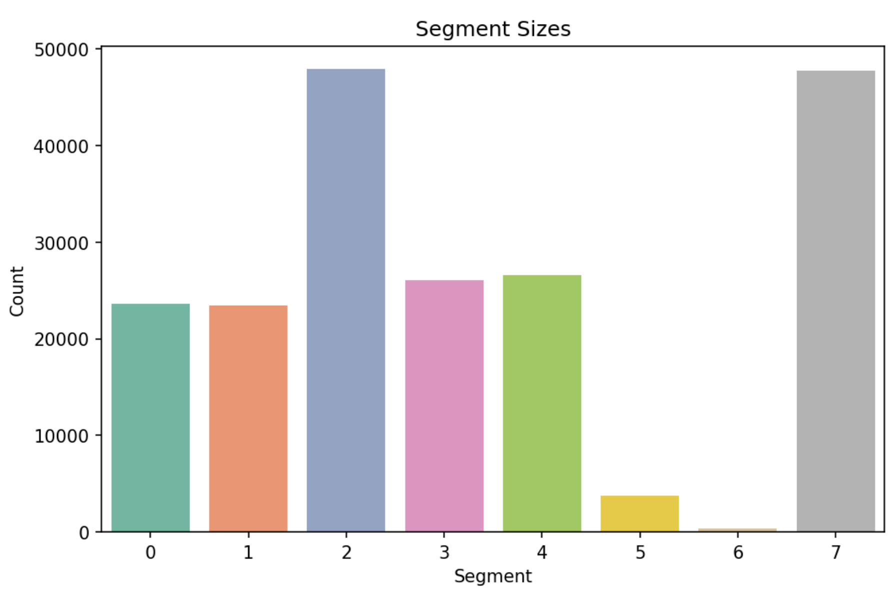

Figure 4. Segment size distribution for k=8 clusters. Segments 2 and 7 dominate, while Segment 6 is small but affluent.

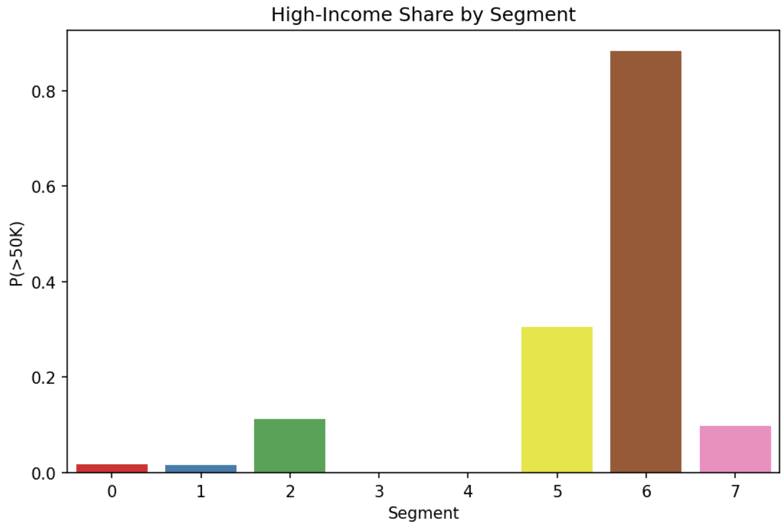

Figure 5. High-income probability by segment. Segment 6 stands out with ~88% >50K; Segments 0–1 are ~2%, Segments 3–4 are ~0%.

Personas & action ideas:

- **Seg 6 (P>50K ≈ 0.88):** Bachelor's+, medical/professional, married males.
  → Premium SKUs, financial services, credit/loyalty, VIP programs.
- **Seg 5 (P>50K ≈ 0.31):** Private sector, pro mix; mid-career.
  → Upsell/cross-sell, aspirational marketing, larger-basket bundles.
- **Seg 2 & 7 (P>50K ≈ 0.10–0.11):** Retail/clerical, HS grads; big segments.
  → Value messaging at scale, promotions, replenishment programs.
- **Seg 0 & 1 (P>50K ≈ 0.02):** Not in labor force; retirees/homemakers.
  → Senior-friendly products/services; deprioritize premium.
- **Seg 3 & 4 (P>50K = 0.00):** Children.
  → Exclude from adult targeting; if household-level, market to guardians.

Notes. Silhouette is modest; that's fine here. The point isn't perfect separation but useful differentiation for messaging and test design. Segments align with labor force status, education, and occupational structure, which is what I want.

5) Evaluation Procedure & Interesting Findings

Procedure recap:

- Train/test split with stratification and fixed seed (reproducible).
- CPS weights included in both training and metrics (population-representative).
- For classification: accuracy, ROC-AUC, PR-AUC, precision, recall, F1, confusion matrix.
- For clustering: silhouette selection on a sample; full fit; summaries and personas.

Findings that mattered:

1. **Label normalization was necessary** ('- 50000.' vs '50000+.'). This could easily trip a pipeline; I put the mapping into data_utils.py.
2. **“Not in universe” is signal**, not noise. It clearly surfaces non-labor-force groups, which explains the clear 0% income segments (children) and low-income retiree segments.
3. **Model behavior matches economics.** High precision for >50K and moderate recall is an acceptable starting point for **cost-aware** marketing. Threshold tuning can rebalance if the campaign goal requires higher reach.
4. **Small high-income niche exists** (Seg 6). It’s small (392 in this dataset) but extremely high P>50K (~0.88). That’s the logical start for premium targeting.
5. **Large value segments** (Seg 2/7) justify scale campaigns; I would not pursue premium conversion there, but I would push value and retention.

6) Business Decisions & Model Usage Recommendations

Why I didn’t oversample/undersample.
Because Walmart’s planning depends on realistic segment sizes and probabilities. If I oversampled >50K to 50/50, I’d inflate the apparent addressable market and distort ROI models. Using CPS weight preserves reality. If recall must increase, I’ll tune the classification threshold and weigh the precision/recall trade-off against campaign economics.

How to use the classifier.

- Score individuals with the saved pipeline.
- Choose a threshold based on business costs/benefits (e.g., higher threshold for premium campaigns to keep precision high; lower threshold when reach is paramount).
- Review calibration if class weighting is introduced later.

How to use the segments.

- Treat them as messaging clusters, not rigid rules.
- Prioritize Seg 6 for premium offers; Seg 5 for aspirational/financing; Seg 2/7 for value promotions; deprioritize children/retirees for premium.
- Run A/B tests by (segment × classifier-score band) to measure lift and refine creative.

Guardrails.

- Review fairness/ethics around sensitive attributes (sex, race, citizenship). I did not drop them here because this is a historical academic dataset; in production I would restrict, report disaggregated metrics, or both.
- Monitor drift if moving to modern data (feature distributions change).

7) Limitations

- **Vintage data (1994–95).** Useful for the exercise but not reflective of current labor markets or prices; treat absolute rates with caution.
- **Imbalance limits recall.** This is expected; threshold tuning can address business needs.
- **Clustering modest separation.** Silhouette ~0.22 is normal for this problem; segments are still actionable for messaging rather than strict policy.
- **No deep feature engineering.** Intentional choice for clarity; can be extended if needed.

8) Future Work

- **Threshold tuning & calibration** to align precision/recall with campaign costs and LTV.
- **Alternative clustering** (Gaussian Mixtures, hierarchical) and supervised segmentation (trees) for rule-based personas.
- **Feature reduction** for very wide OHE (target encoding with CV, numeric PCA, category grouping).
- **Modern data refresh** and live A/B testing to measure real lift.
- **Fairness diagnostics** on any deployment dataset.

9) Reproducibility

- Code lives in src/ with small, readable modules:
    - train_classify.py, train_segments.py, data_utils.py, preprocess.py, report_utils.py, visualize.py.
- Commands and environment are in README.md (PowerShell).
- Outputs saved under results/. Human-readable artifacts (*.json, *.csv, *.md) are committed for easy review; heavy binaries are git-ignored.

10) References

- **Scikit-learn User Guide** (preprocessing, model evaluation, metrics).
- **LightGBM Documentation** (LGBMClassifier, parameters, handling imbalance).
- **U.S. Census Bureau - Current Population Survey** documentation (universe definitions, weighting, coding notes).

11) Notes to Stakeholders

- Do we plan to target households or individuals? If households, I would re-aggregate by household ID and re-derive features.
- What are the campaign economics (cost per contact, expected margin uplift)? I can set a score threshold that maximizes expected profit.
- Which channels (email, app, in-store) and privacy constraints should we respect for real deployment?
- Are there fairness constraints we should hard-enforce (e.g., drop certain features, or monitor disparities)?

Appendix: Key Files

- src/train_classify.py — training + evaluation, saves best_classifier.pkl and metrics
- src/train_segments.py — clustering + personas, saves segments_summary.csv & segment_personas.md
- src/data_utils.py — load columns/data, label normalization, split features/target/weights
- src/preprocess.py — one-hot + scaling ColumnTransformer
- src/report_utils.py — small save helpers (UTF-8, Windows-friendly)