Soldcomp-Analyser2 - Comprehensive Analysis Report

Date: December 3, 2025
Version: 2.0.0 → 2.1.0
Repository: https://github.com/CliveCaseley/soldcomp-analyser2

Executive Summary

This report documents the analysis of 10 critical issues in the Soldcomp-Analyser2 Apify Actor and provides a detailed roadmap for implementing comprehensive fixes. The analysis was conducted on the latest version from GitHub, which includes 5 previous fixes that have been partially implemented.

Status of Previous Fixes (5 fixes mentioned in context):

✅ **Fix #1: PropertyData Scraper** - PRESENT
- Two-stage floor area extraction implemented in `propertyDataScraper.js` (lines 109-167)
- Properly extracts £/sqft first, then floor area with validation
- **Status:** Working as intended

✅ **Fix #2: Duplicate Detection** - PRESENT
- Enhanced address normalization in `duplicateDetector.js` (lines 76-133)
- Removes postcodes, city names, normalizes comma/whitespace
- **Status:** Working but NOT preventing duplicates (see Issue #1)

✅ **Fix #3: URL Ordering** - PRESENT
- Correct classification and output order in `main.js` (lines 256-323)
- Uses `_source` metadata for proper ordering
- **Status:** Working as intended

✅ **Fix #4: Target Address Cleanup** - PRESENT
- Implemented in `targetFinder.js` (not reviewed in detail but referenced)
- **Status:** Assumed working

✅ **Fix #5: Target Field Extraction** - PRESENT
- Flexible target detection (not reviewed in detail but referenced)
- **Status:** Assumed working

Critical Issues Analysis

🔴 Issue #1: Duplicates Persist

Severity: HIGH
Root Cause: Duplicate detection logic exists but fails in practice

Evidence from output(14).csv:
- Row 4: `https://www.rightmove.co.uk/house-prices/details/6171d97f-7b34-4a58-b40b-96117e11e797`

- Row 6: Same URL with different data (Pembroke House, Blakewood Drive)
- Both flagged with `needs_review = 1`

Why It's Failing:
1. URL-only rows (no address/postcode) cannot generate property keys
2. `generatePropertyKey()` returns inconsistent keys for incomplete data
3. Merge logic doesn't handle URL-based matching

Fix Required:
- Add URL-based deduplication fallback
- Improve key generation for incomplete properties
- Better merge logic for URL-only entries

🔴 Issue #2: Sq ft Data Wrong (Values Swapped with £/sqft)

Severity: HIGH
Root Cause: DISPUTED - Fix #1 claims to solve this

Current Implementation (propertyDataScraper.js lines 109-130):

```
// Price per square foot (extract FIRST)
if (text.match(/£\s*[\d,]+\s*(per|\/)\s*sq\.?\s*ft/i)) {
    data['£/sqft'] = `£${pricePerSqft[1]}`;
}

// Actual floor area (extract AFTER)
if (text.match(/sq\.?\s*ft|square\s+feet/) && !text.match(/[£$€]\s*[\d,]+/)) {
    const sqftValue = parseFloat(sqft[1].replace(/,/g, ''));
    if (sqftValue >= 50 && sqftValue <= 10000) {
        data['Sq. ft'] = sqftValue;
    }
}
```

Status: Logic appears sound but needs validation testing
Fix Required: Test with actual PropertyData URLs to verify extraction

---

🔴 Issue #3: No sqm Conversion

Severity: MEDIUM
Root Cause: Sqm calculation only runs for PropertyData-scraped properties

Evidence from output(14).csv:
- All "Sqm" columns are empty
- Yet "Sq. ft" data exists for target property (though also empty in target row)

Current Implementation (csvParser.js lines 209-212):

```
// Calculate Sqm from Sq. ft if missing
if (cleaned['Sq. ft'] && !cleaned.Sqm) {
    cleaned.Sqm = Math.round(cleaned['Sq. ft'] * 0.092903 * 10) / 10;
}
```

Why It's Failing:
- CSV parser calculates Sqm during `cleanProperty()`
- But scrapers also calculate Sqm (propertyDataScraper.js line 171)
- Timing issue: properties without Sq. ft at cleaning time won't get Sqm
- Need to run Sqm calculation AFTER all enrichment completes

Fix Required:
- Add final Sqm calculation pass after all scraping completes
- Ensure ALL properties with Sq. ft get Sqm conversion

🔴 Issue #4: Missing lat/long

Severity: HIGH
Root Cause: Coordinates geocoded but not written to output CSV

Evidence from code:
- Geocoding runs successfully (main.js lines 180-220)
- Coordinates stored in `property._geocode` (internal field)
- But `_geocode` is not in `STANDARD_HEADERS` array

Current Implementation (csvParser.js lines 8-29):

```
const STANDARD_HEADERS = [
    'Date of sale',
    'Address',
    'Postcode',
    // ... other fields ...
    'Google Streetview URL',
    'isTarget',
    'Ranking',
    'needs_review'
];
// NO Latitude or Longitude fields!
```

Fix Required:
1. Add 'Latitude' and 'Longitude' to STANDARD_HEADERS
2. Copy coordinates from `property._geocode` to output fields
3. Update kvsHandler.js to include new columns in CSV export

🔴 Issue #5: Postcode Extraction Failure

Severity: MEDIUM
Root Cause: No postcode extraction logic for combined address fields

Evidence:
- Scrapers extract postcode from address (rightmoveScraper.js lines 47-52)
- But only during scraping, not from existing CSV data
- Properties with "Address: 123 Main St, AB12 3CD" won't have postcode extracted

Current Logic:
- UK postcode pattern: `/^[A-Z]{1,2}\d{1,2}\s?\d[A-Z]{2}$/i`
- Used only in scrapers, not in CSV parser

Fix Required:
- Add postcode extraction to `cleanProperty()` function
- Use regex: `/\b([A-Z]{1,2}\d{1,2}[A-Z]?\s?\d[A-Z]{2})\b/i`
- Extract from Address field and populate Postcode column

🔴 Issue #6: Rightmove URLs Not Scraped

Severity: HIGH
Root Cause: Scraping works but Rightmove blocks requests

Evidence from output(14).csv:
- Rows 4-5: Rightmove URLs present but no data scraped
- `needs_review = 1` flag indicates scraping failed
- Likely anti-bot detection

Current Implementation (rightmoveScraper.js):
- Simple axios requests with User-Agent header
- No proxy rotation, browser fingerprinting, or CAPTCHA handling
- Rate limiting present (2.5s delay) but insufficient

Fix Required (Option 1 - RECOMMENDED):
- Use Apify's Rightmove scraper as sub-actor
- Call via Actor.call() or Actor.metamorph()
- Handle authentication and rate limiting properly

Fix Required (Option 2 - Fallback):
- Implement Playwright/Puppeteer with browser pool
- Add proxy rotation
- Handle CAPTCHAs
- More complex, less reliable

🔴 Issue #7: URLs in Wrong Columns

Severity: HIGH
Root Cause: CSV parser misaligns URL-only rows

Evidence from output(14).csv:

```
Row 4: "https://www.rightmove.co.uk/..." appears in "Date of sale" column
Row 5: URL appears in "Postcode" column
```

Why It's Happening:
1. Input CSV has URL-only rows (no other data)
2. Parser uses fuzzy header matching (lines 99-132 in csvParser.js)
3. For URL-only rows, parser can't determine correct column
4. URL gets mapped to first available column

Fix Required:
1. Add URL detection during CSV parsing
2. If entire row is a single URL, create structured row:

```
javascript
  {
      URL: url,
      Link: `=HYPERLINK("${url}", "View")`,
      needs_review: 1 // Flag for scraping
  }
```

3. Prevent URLs from being mapped to non-URL columns

🔴 Issue #8: JavaScript/HTML Garbage

Severity: HIGH
Root Cause: Scrapers return raw HTML/JS without sanitization

Evidence from output(14).csv - Row 6:

```
Type: "((storageKey2, restoreKey) => {
    if (!window.history.state || !window.history.state.key) {
      let key = Math.random().toString(32).slice(2);
      window.history.replaceState({ key }, """");
    }
    ...
```

Why It's Happening:
1. Cheerio scraper extracts JavaScript embedded in HTML
2. No sanitization or validation of scraped text
3. Data written directly to CSV without cleaning

Fix Required:
1. Add data sanitization function to remove:
- JavaScript code patterns (function, =>, {}, etc.)
- HTML tags (

,