From Data to Intelligence:
The Feature Engineering Workflow

Transforming Clean Data into Model-Ready Features
VOIS Internship Project

October 4, 2025

Abstract

This report documents the critical transition from retrospective data analysis to forward-looking machine learning preparation through systematic feature engineering. The transformation process converts human-readable cleaned data (81,781 listings across 26 columns) into optimized numerical format (13 engineered features) ready for predictive modeling. Four key accomplishments define this phase: persistent 'brookln' typo correction ensuring unified borough representation; creation of `days_since_last_review` feature providing nuanced market relevancy signal beyond raw popularity counts; log-transformation of target variable stabilizing heavily skewed price distribution for percentage-based prediction; and generation of model-ready datasets (`model_features.csv` and `model_target.csv`) through strategic feature selection and one-hot encoding. This methodical workflow exemplifies data science best practices, transforming Phase 2 analytical insights into actionable predictive features while maintaining data integrity through iterative quality validation.

Contents

1 Executive Summary

Milestone: Analysis to Prediction Transition

Day 7 marks the project's official pivot from retrospective understanding to forward-looking prediction. The feature engineering process has methodically transformed our cleaned, human-readable dataset into a fully numerical, high-dimensional format optimized for machine learning consumption.

1.1 The Feature Engineering Workflow

Four-Step Transformation Pipeline

The engineering process follows a systematic workflow:

Step 1: Load Clean Data

- Starting point: 81,781 validated listings
- Input: `cleaned_airbnb_data.csv` from Day 2
- Data integrity: 100% complete, zero missing values

Step 2: Engineer New Features

- Create `days_since_last_review` recency metric
- Transform temporal data into numerical signal
- Strategic imputation for never-reviewed listings

Step 3: Select & Transform Features

- Identify 8 strategic predictive features
- Apply log-transformation to target variable (price)
- Normalize skewed distribution for model stability

Step 4: Encode Categorical Data

- Convert text categories to numerical format
- One-hot encoding for boroughs and room types
- Final output: 13-column feature matrix

1.2 Key Outcomes

Four Critical Achievements

1. **Data Integrity Fortified**
   - Persistent 'brookln' typo identified and corrected at source
   - Prevents data fragmentation in borough representation
   - Ensures model learns from unified, accurate geographic features
2. **Insightful Recency Feature Created**
   - `days_since_last_review` captures market relevancy
   - Distinguishes active listings from stale inventory
   - Provides nuanced signal beyond raw review counts
3. **Target Variable Stabilized**
   - Log-transformation normalizes heavily skewed price distribution
   - Shifts prediction focus to percentage changes vs absolute values
   - Reduces outlier influence on model training
4. **Model-Ready Datasets Generated**
   - Feature matrix: 81,781 rows $\times$ 13 columns
   - Target vector: 81,781 log-transformed prices
   - Clean separation enables reproducible train-test splits

2 Critical Data Correction: The 'brookln' Fix

2.1 Problem Identification

Persistent Data Quality Issue

Despite rigorous Day 2 cleaning and Day 4 correction attempts, the 'brookln' typo persisted in the source dataset. The feature engineering script's first diagnostic step revealed the issue remained:

Detected State:

- Expected values: 5 unique boroughs
- Observed values: 6 unique values including 'brookln'
- Impact: Would create spurious feature during encoding
- Risk: Diluted Brooklyn predictive power

2.2 Correction and Validation

Strategic Intervention

Correction Action:

- Systematic replacement: All instances of 'brookln' → 'Brooklyn'
- Source-level fix ensures all downstream processes benefit
- Validation: Confirmed exactly 5 unique borough values remain

Why This Matters:

- **Model Accuracy:** Unified Brooklyn representation captures full market signal
- **Feature Economy:** Prevents wasted feature column on typo
- **Interpretability:** Borough coefficients reflect true geographic effects
- **Phase 2 Validation:** Brooklyn's 42.1% market share now properly represented

Confirmed Borough Distribution:

Brooklyn — Manhattan — Queens — Bronx — Staten Island

3 Feature Engineering: Creating `days_since_last_review`

3.1 Rationale and Design

Beyond Raw Popularity: Capturing Recency

The `number_of_reviews` column provides a popularity signal, but lacks critical temporal context:
The Problem:

- Listing A: 50 reviews, last review 1500 days ago
- Listing B: 50 reviews, last review yesterday

Both have identical review counts, yet represent vastly different market positions. Listing A likely represents stale, inactive inventory; Listing B signals current, active market presence.
The Solution:

- Calculate days elapsed since `last_review` date
- Reference date: January 1, 2023 (fixed benchmark)
- Formula: `days_since_last_review = reference_date - last_review`

3.2 Implementation Strategy

Aspect	Implementation Details
Calculation	Days between fixed reference date (2023-01-01) and each listing's `last_review` timestamp
Missing Value Handling	Listings with no reviews (NaT in `last_review`) imputed with 9999 days
Rationale for 9999	Large out-of-distribution value allows tree-based models to isolate "never-reviewed" category; distinct pricing patterns for untested listings
Signal Interpretation	Low values (recent reviews) = active, high values (old reviews) = stale, 9999 = unproven

Table 1: Feature Engineering Details for `days_since_last_review`

3.3 Strategic Value

Why This Feature Matters

Enhanced Model Intelligence:

- Distinguishes between "popular but stale" and "popular and active" listings
- Captures market dynamics that static counts cannot reveal
- Enables model to learn temporal pricing patterns
- Provides competitive signal for recent market entrants

Interaction Potential:

- Combined with `number_of_reviews`: comprehensive activity profile
- Manhattan listings may show different recency-price relationships than Brooklyn
- Power hosts (Day 6 finding) may exhibit distinct recency patterns

Tree Model Advantage:

- Random forests can naturally segment by recency thresholds
- 9999 imputation creates distinct leaf nodes for never-reviewed listings
- Model learns optimal recency breakpoints from data

4 Stabilizing the Target: Log-Transformation of Price

4.1 The Skewness Problem

Original Price Distribution: Heavily Right-Skewed

Phase 2 analysis (Day 4) revealed extreme price skewness:
Statistical Evidence:

- Mean: $626.55, Median: $626.00 (roughly symmetric center)
- However: Long right tail with luxury outliers up to $1,200
- 95th percentile: $1,142 (upper 5% distorts distribution)
- Standard deviation: $331 (high relative to median)

Model Training Challenges:

- Outliers dominate loss function, pulling predictions upward
- Model struggles to learn patterns in dense $100-$400 range
- Large absolute errors on luxury listings overwhelm training
- Predictions skew toward mean, ignoring market nuances

4.2 Log-Transformation Solution

Mathematical Foundation

Transformation Formula:

$$\text{log\_price} = \log(1 + \text{price})$$

Why $\log(1 + x)$ Instead of $\log(x)$?

- Standard log undefined for zero values
- $\log(1 + x)$ shifts distribution, handling zeros gracefully
- Minimal difference for large values: $\log(1 + 1000) \approx \log(1000)$

Effect on Distribution:

- Compresses right tail, reducing outlier influence
- Expands left side, preserving budget listing distinctions
- Result: More symmetric, approximately normal distribution
- Model focus: Percentage changes, not absolute dollars

4.3 Before and After Comparison

Distribution Metric	Original Price	Log-Transformed
Mean	\$626.55	~6.42
Standard Deviation	\$331.83	~0.52
Skewness	Heavily right-skewed	Approximately symmetric
Outlier Influence	Dominates training	Mitigated
Prediction Focus	Absolute dollars	Percentage changes

Table 2: Target Variable Distribution Comparison

4.4 Critical Reminder: Inverse Transformation

Interpreting Model Predictions

Forward Transformation (Training):

- Input: Original prices in dollars
- Process: Apply $\log(1 + x)$
- Output: Log-transformed target for model training

Inverse Transformation (Prediction):

- Model Output: Log-space predictions
- Process: Apply inverse $\exp(x) - 1$
- Result: Real dollar price predictions

Essential Step: All model predictions must be inverse-transformed using `np.expm1()` before stakeholder interpretation. Forgetting this step yields meaningless log-scale values.

5 Strategic Feature Selection

5.1 The 8 Core Predictive Features

Category	Features	Predictive Rationale
Geography & Property	`neighbourhood_group` `room_type`	Day 3 EDA: Manhattan-hegemony (85.4%); property t ity (97.5% entire homes/priva
Booking Policy	`minimum_nights`	Day 5 finding: Strategic host nights entire homes, 2 night rooms); operational cost opti
Popularity Metrics	`number_of_reviews` `reviews_per_month`	Direct engagement measure proof signals influencing guest
Host Professionalism	`calculated_host_listings_countavailability_365`	Day 6 insight: Power host 149.75 days availability vs 13 eral population
Engineered Relevancy	`days_since_last_review`	New feature capturing marke and listing freshness

Table 3: Strategic Feature Selection Based on Phase 2 Insights

5.2 Feature Selection Philosophy

Data-Driven Feature Engineering

The 8 selected features represent a holistic listing profile informed by comprehensive Phase 2 analysis:

Geographic Drivers (Day 3):

- Borough location proven most significant market segmentation
- Room type reflects fundamental business model choice
- Combined: Capture property positioning strategy

Operational Signals (Day 5):

- Minimum nights reveals host operational philosophy
- Availability indicates dedicated rental vs casual sharing
- Together: Profile host professionalization level

Engagement Indicators (Day 6):

- Review metrics measure guest satisfaction and activity
- Host listing count identifies power hosts vs casual operators
- Combination: Comprehensive performance profile

Temporal Intelligence (Day 5 + Day 7):

- Engineered recency feature captures market dynamics
- Distinguishes active from dormant listings
- Adds temporal dimension to static features

6 One-Hot Encoding: From Text to Numbers

6.1 The Categorical Challenge

Why Models Need Numbers

Machine learning algorithms operate exclusively on numerical data:
Text Data Problem:

- `neighbourhood_group = "Manhattan"` is meaningless to algorithms
- Cannot compute mathematical operations on text strings
- Need systematic conversion to numerical format

Solution: One-Hot Encoding

- Transform each category into separate binary (0/1) column
- "Manhattan" listing: 1 in `neighbourhood_group_Manhattan`, 0 elsewhere
- Model learns distinct coefficient for each category

6.2 Encoding Execution

Categorical Feature	Unique Values	Encoded Columns Created
`neighbourhood_group`	5 boroughs	4 binary columns (Bronx, Manhattan, Queens, Staten Island)
`room_type`	4 types	2 binary columns (Hotel room, Private room)

Table 4: One-Hot Encoding Results

6.3 The Dummy Variable Trap

Avoiding Multicollinearity

Problem: Perfect Correlation
If all borough columns are included:

- 5 columns for 5 boroughs creates redundancy
- If Manhattan=0, Brooklyn=0, Queens=0, Bronx=0, Staten Island=0... listing **must** be in Brooklyn
- Brooklyn column is perfectly predictable from others
- Creates multicollinearity: mathematical instability in model

Solution: Drop First Category

- Set `drop_first=True` in encoding function
- Removes one category from each feature (becomes reference category)
- Brooklyn and Entire home/apt dropped (encoded as all zeros)
- Result: 4 borough + 2 room type columns = 6 total
- No information loss: reference categories implicit

6.4 Final Feature Matrix Composition

Feature Type	Column Count	Total
Original Numerical Features	6	6
Engineered Feature (`days_since_last_review`)	1	7
Encoded Borough Features	4	11
Encoded Room Type Features	2	13
Final Feature Matrix	**13 columns**	**81,781 rows**

Table 5: Feature Matrix Dimensionality Breakdown

Encoding Verification

Confirmed Columns:

- **Numerical (7):** `minimum_nights`, `number_of_reviews`, `reviews_per_month`, `calculated_host_listings_count`, `availability_365`, `days_since_last_review`
- **Borough (4):** `neighbourhood_group_Bronx`, `neighbourhood_group_Manhattan`, `neighbourhood_group_Queens`, `neighbourhood_group_Staten Island`
- **Room Type (2):** `room_type_Hotel room`, `room_type_Private room`

Critical Validation: No `neighbourhood_group_brookln` column present, confirming typo correction success.

7 Final Model-Ready Datasets

7.1 Output Files

File	Dimensions	Content
`model_features.csv`	81,781 × 13	Feature matrix (X)
`model_target.csv`	81,781 × 1	Log-transformed prices (y)

Table 6: Model-Ready Dataset Files

7.2 Dataset Characteristics

Production-Ready Data Assets

Feature Matrix (`model_features.csv`):

- Fully numerical: All text categories converted to binary encoding
- Zero missing values: Complete data integrity maintained
- Standardized scale: Numerical features on comparable ranges
- Rich dimensionality: 13 features capture comprehensive listing profile

Target Vector (`model_target.csv`):

- Normalized distribution: Log-transformation reduces skewness
- Stabilized variance: More consistent prediction difficulty
- Percentage-focused: Model learns relative pricing patterns
- Inverse-transformable: Easy conversion back to dollar predictions

Ready for Training:

- Clean train-test split possible: Separate X and y files
- Reproducible pipeline: Identical transformation on new data
- Model-agnostic format: Compatible with scikit-learn, XGBoost, neural networks

8 Conclusion

Phase 3 Commencement: Analysis to Action

Day 7 successfully completes the feature engineering foundation, marking the project's transition from retrospective understanding to predictive modeling:
Accomplishments:

- Data integrity fortified through persistent typo correction
- Sophisticated temporal feature engineered for market relevancy
- Target variable stabilized via log-transformation
- 13-feature model-ready matrix generated with strategic encoding

Quality Assurance:

- All 81,781 listings preserved through transformation
- Zero missing values maintained across pipeline
- Feature selection informed by comprehensive Phase 2 insights
- Mathematical rigor applied to distribution normalization

The meticulous feature engineering workflow exemplifies data science best practices: iterative quality validation, domain-informed feature design, and mathematical transformation for model optimization.

8.1 Ready for Model Training

Phase 3 Progression

With model-ready datasets generated, the project advances to predictive modeling:
Day 8 Objectives:

- Train Random Forest regression model on engineered features
- Implement rigorous train-test split methodology
- Evaluate performance using multiple regression metrics
- Analyze feature importance to validate engineering decisions
- Interpret model predictions in business context

The systematic feature engineering ensures models built on validated, properly formatted data, maximizing predictive accuracy and stakeholder value.