- **Legacy systems** already built on SVM infrastructure
- Academic exercises or when comparing to historical baselines

For production book pricing systems, SVMs have been largely supplanted by gradient boosting (for structured data) and deep learning (for unstructured data). They remain pedagogically useful but are no longer the practical choice for new implementations.

2.5 Deep Learning: Neural Networks for Multimodal Data

Deep neural networks offer a powerful framework for incorporating unstructured inputs (text descriptions, cover images) alongside structured metadata. They can learn complex, hierarchical representations that capture nuanced patterns in high-dimensional data.

2.5.1 Architecture Foundations

A basic **feedforward neural network** for price prediction might look like:

Input Layer: Numeric features (year, ratings, num_pages) + Embeddings (genre, author) + Text features
↓
Hidden Layer 1: 128 neurons, ReLU activation
↓
Hidden Layer 2: 64 neurons, ReLU activation
↓
Dropout (0.3): Regularization
↓
Output Layer: 1 neuron, linear activation → predicted price

Key components:

- **Embeddings**: Convert categorical features (author ID, publisher, genre) into learned dense vectors (e.g., 10-50 dimensions). The network learns that similar authors should have similar embeddings.
- **Activation functions**: ReLU (Rectified Linear Unit) is standard for hidden layers, enabling nonlinearity
- **Dropout**: Randomly zeros out neurons during training to prevent co-adaptation (overfitting)
- **Batch normalization**: Normalizes activations between layers, stabilizing training

2.5.2 Text-Based Pricing Models

Recent approaches treat price prediction as an **NLP problem**, feeding textual product descriptions directly into sequence models.

Text2Price framework (Saraçlar et al., 2022):

- Concatenates product title, category, and description into a single text sequence
- Feeds into either an **LSTM** (Long Short-Term Memory) or **CNN** (Convolutional Neural Network)
- Outputs a price prediction

Empirical results:

- **LSTM-based model**: 6.16% MAPE (Mean Absolute Percentage Error)
- **CNN-based model**: 7.14% MAPE
- **Training time**: CNN was 3× faster but less accurate

The LSTM's sequential processing better captured the order-dependent semantics in product descriptions (e.g., "new, never opened" vs. "opened, like new"). The CNN's parallel convolutions were faster but lost some subtle context.

Interpretation: 6.16% MAPE means the model's predictions were typically within **±6%** of true prices—excellent for a general marketplace model spanning many product categories.

2.5.3 Multimodal Deep Learning

Combining text and images can significantly improve predictions, especially for used items where visual condition matters.

Fathalla et al. architecture:

Image Input → CNN (ResNet/VGG) → Image Features (512-dim)
↓
Text Input → LSTM → Text Features (256-dim) → Concatenate → Dense Layers → Price + Range

Key findings:

- The hybrid model produced "promising results" and could output both a **point estimate** and a **price range** (minimum and maximum expected price)
- Image features helped assess condition for categories where text descriptions were sparse
- The model learned that certain visual patterns (wear, discoloration, bent covers for books) correlated with lower prices

Limitations:

- Requires **large datasets** (typically 50k+ samples) to avoid overfitting
- **Computationally expensive** (GPU required for reasonable training times)
- **Less interpretable** than tree models (though SHAP and attention visualization help)
- Images add data collection complexity (many listings lack quality photos)

2.5.4 Transformer-Based Approaches

BERT and modern transformers can be fine-tuned for regression tasks on book-related text.

Example pipeline:

1. Concatenate book title, synopsis, and metadata into a text sequence
2. Tokenize with BERT tokenizer (WordPiece, max 512 tokens)
3. Feed through pre-trained BERT (or DistilBERT for efficiency)
4. Take `[CLS]` token embedding (768-dim representation)
5. Add dense layers for regression: `CLS → Dense(128) → Dense(1) → Price`

Empirical results: One study using DistilBERT embeddings of book synopses (6k books dataset) improved error metrics **10-15%** compared to simpler text encoding (TF-IDF). The pre-trained language model captured semantic relationships (e.g., "fantasy epic" and "sword and sorcery" are related genres, commanding similar price premiums).

Advantages over CNNs/LSTMs:

- **Transfer learning**: BERT was pre-trained on billions of words, bringing general language understanding
- **Bidirectional context**: Understands full sequence context vs. LSTMs' left-to-right processing
- **Subword tokenization**: Handles rare words better via WordPiece decomposition

Disadvantages:

- **Computationally heavy**: DistilBERT has 66M parameters; full BERT has 110M
- **Requires fine-tuning**: Can't just use off-the-shelf embeddings; must train on pricing task
- **Diminishing returns**: For simple text (eBay titles like "Harry Potter Hardcover Good Condition"), may not outperform TF-IDF + XGBoost

2.5.5 When Deep Learning Excels

Neural networks are particularly valuable when:

1. **Large datasets available** (>50k samples): Enough data to learn complex patterns without overfitting
2. **Unstructured data dominates**: Text descriptions, cover images, or user reviews are primary signals
3. **Multimodal inputs**: Combining text, images, and structured features in a unified architecture
4. **Transfer learning applicable**: Pre-trained models (BERT, ResNet) provide strong initialization
5. **Latent patterns exist**: Subtle semantic relationships in text that keyword matching would miss

Where tree ensembles still win:

- **Small to medium datasets** (< 50k samples)
- **Primarily structured data** (edition, condition, ratings as categorical/numeric features)
- **Need for interpretability** (feature importance, explaining predictions to users)
- **Limited computational resources** (no GPU required for XGBoost)

2.5.6 Hybrid Deep Learning + Boosting

An emerging **best practice** combines the strengths of both approaches:

Two-stage pipeline:

1. **Stage 1 (Deep Learning)**: Extract rich representations from unstructured data
    - Run BERT on book synopsis → 768-dim embedding
    - Run ResNet on cover image → 512-dim embedding
    - Run sentiment analysis on reviews → polarity scores
2. **Stage 2 (Gradient Boosting)**: Combine with structured features
    - Concatenate: [BERT embedding, image embedding, sentiment, edition, condition, ratings, ...]
    - Train XGBoost/LightGBM on this enriched feature set

Why this works:

- Deep learning handles the hard parts (understanding "this gripping tale of redemption" indicates literary fiction)
- Boosting excels at the structured parts (condition="Good" → 30% discount, latest edition → premium)
- Boosting is robust to the high-dimensional deep learning features
- Can use smaller neural networks (less overfitting risk) since boosting will do final integration

Practical example: A Kaggle team for Mercari used:

- Word2Vec embeddings (50-dim) for item names and descriptions
- Category embeddings (10-dim) from a small neural network
- Fed these + structured features into LightGBM
- **Result**: Top 5% on leaderboard with manageable training time

This hybrid approach is increasingly the **industry standard** for marketplace pricing, offering:

- **Better accuracy** than either approach alone
- **Moderate complexity** (not a full end-to-end deep network)
- **Easier debugging** (can inspect boosting feature importance)
- **Faster iteration** (can retrain boosting part without re-running deep models)

2.6 Ensemble and Hybrid Models: Combining Everything

In competitive environments (Kaggle, production A/B tests), the best results typically come from **ensembling multiple diverse models**.

2.6.1 Types of Ensembles

Simple averaging/voting:

```
Final_Price = (Model1_Pred + Model2_Pred + Model3_Pred) / 3
```

- Works when models are comparable in quality
- Reduces variance by averaging out individual model errors
- No additional training required

Weighted averaging:

```
Final_Price = w1*Model1 + w2*Model2 + w3*Model3  where Σw = 1
```

- Weights determined by validation performance
- Gives more influence to better models
- Can use optimization (e.g., minimize RMSE on validation set to find optimal weights)

Stacking (meta-learning):

```
Stage 1: Train base models (Linear, RF, XGBoost, LSTM) on training data
Stage 2: Use their predictions as features for a meta-model
```

Example:

1. Train 5 diverse models, get their predictions on validation set
2. Train a **meta-model** (often linear regression or small boosting model) that takes `[pred1, pred2, pred3, pred4, pred5]` as input, outputs final price
3. The meta-model learns when to trust which base model

Why stacking works:

- Different models make different types of errors
- Linear model might underfit but captures global trends
- Deep model might overfit but captures text nuances
- XGBoost is a good all-arounder
- Meta-model learns to weight them based on input characteristics

Empirical evidence: Zhu et al. (2024) demonstrated a "multi-model output fusion" integrating:

- Linear regression (captures linear trends)
- Decision tree (interpretable interactions)
- Gradient boosting (accurate complex patterns)

Their ensemble achieved **lower MAE, lower RMSE, and higher R^2** than any individual model, with an "exceptionally high R^2" on test data. The diversity of approaches meant that when one model struggled (e.g., linear model on nonlinear edge cases), others compensated.

2.6.2 Diversity is Key

Why not just ensemble many XGBoost models?

- High correlation between similar models → limited ensemble benefit
- Better to combine fundamentally different approaches

Strategies for diversity:

1. **Different algorithms**: Linear + Tree + Neural
2. **Different feature sets**: One model uses only metadata, another uses text, another uses both
3. **Different data samples**: Bagging (different bootstrap samples)
4. **Different hyperparameters**: XGBoost with depth=3 vs. depth=10
5. **Different representations**: One model uses TF-IDF, another uses BERT embeddings

Diminishing returns:

- 2-3 diverse models: Large improvement over single model
- 4-7 models: Moderate additional gains
- 8+ models: Marginal improvement, increased complexity often not worth it for production

2.6.3 Production Considerations

Complexity vs. Benefit:

- A 10-model ensemble might improve RMSE by 5% over the best single model
- But it requires 10× inference time, 10× maintenance burden, 10× things that can break
- Often better to invest in better features or more training data for the best single model

When to ensemble in production:

- **High-value predictions** (e.g., dynamic pricing for high-volume sellers where small improvements = large revenue)
- **Low latency requirements** (can't run 10 models per request on mobile app)
- **Critical accuracy** (fraud detection, medical pricing)

Practical compromise:

- Use ensemble for **model development** to push performance limits
- **Distill** the ensemble into a single model (train a single XGBoost to mimic the ensemble's predictions)
- Deploy the distilled model for speed while retaining most of the accuracy

---

3. Data Sources and Key Features for Book Pricing

A pricing model is only as good as its data. This section covers where to obtain training data and which features drive predictive power.

3.1 Public Datasets and Competitions

MachineHack "Predict the Price of Books" Challenge

- **Size**: 6,237 books (training) + 1,560 (test)
- **Features**: Title, Author, Edition, Reviews, Ratings, Synopsis, Genre, Category, Price
- **Coverage**: Broad mix of genres, authors, and price ranges
- **Quality**: Carefully curated with clean labels

Key learning: This dataset demonstrates the value of **text features**. Top participants extracted:

- Topics from synopsis using Latent Dirichlet Allocation (LDA)
- TF-IDF vectors from titles
- Edition year and format by parsing the Edition string ("Paperback – Import, 26 Apr 2018")
- Author popularity metrics

Winners achieved **RMSLE < 0.20** (roughly ±20% error), showing that with rich features and good modeling, strong accuracy is achievable.

Kaggle Mercari Price Suggestion Challenge

- **Size**: 1.4 million marketplace listings
- **Features**: item_name, category, brand, condition, shipping, item_description, price
- **Domain**: General marketplace (not book-specific) but highly relevant
- **Key preprocessing**: Organizers removed price mentions from descriptions (`[rm]`) to prevent leakage

Critical insights:

1. **Text matters enormously**: Models using `item_description` improved significantly over those using only structured features

2. **High cardinality categoricals**: Brand had ~4,500 unique values → required smart encoding (target encoding, embeddings)
3. **Skewed target**: Prices ranged from $3 to $2,000+ → log transformation essential
4. **Evaluation metric**: RMSLE (Root Mean Squared Log Error) to reduce outlier impact

Winning approaches:

- Blended models: Gradient Boosting (LightGBM, XGBoost) + Neural Networks (LSTM, CNN on text) + Factorization Machines
- Feature engineering: Character-level n-grams, word embeddings, category tree encoding
- Ensembles: Top team used 20+ models with diverse architectures

Goodreads Dataset (Kaggle)

- **Size**: ~38,000 books
- **Features**: Ratings, number of ratings, publication year, author, genres, often price proxies
- **Strength**: Rich social data (user ratings, reviews)
- **Limitation**: Not all books have resale prices attached

Usage pattern: Combine with marketplace data (eBay/Amazon) to augment popularity features.

3.2 eBay Data Collection

eBay provides abundant training data via **completed listings** and **sold items** feeds.

Data Sources

1. **eBay API** (official):
    - Finding API: Search completed/sold listings for specific queries (e.g., ISBN, title)
    - Returns: Final price, sale date, condition, title, seller info, listing format (auction/BIN)
    - **Limitations**: Rate limits (5,000 calls/day for basic tier), historical data limited to 90 days
2. **Web scraping** (advanced sold listings):
    - eBay's website shows more historical data than API
    - Can extract: Full descriptions, number of photos, HTML formatting details
    - **Legal/ethical**: Must respect robots.txt and Terms of Service; consider data use policies
3. **Academic datasets**:
    - Ghani & Simmons (2004/2008) collected historical auction data for research
    - Studies like Bodoh-Creed et al. compiled 143+ auctions for specific items
    - Often available by request for research purposes

Key Features from eBay Listings

Item-specific:

- **Title**: Often contains critical info (edition, condition keywords like "mint", "water damage", "signed")
- **Condition**: New, Like New, Very Good, Good, Acceptable, For Parts
- **Item specifics**: Publisher, publication year, format (hardcover/paperback), ISBN
- **Description**: Full text description (may include detailed condition notes, provenance)
- **Photos**: Number of images, whether stock photo used vs. actual item

Listing mechanics:

- **Format**: Auction vs. Buy It Now (fixed price)
- **Starting price**: For auctions (correlates with final price but can introduce bias)
- **Duration**: 1, 3, 5, 7, or 10-day auction
- **Ending time**: Day of week and time (Sunday evening auctions often get higher prices)
- **Shipping cost**: High shipping can depress bids (buyers consider total cost)
- **Returns accepted**: Reduces buyer risk, may increase price

Seller features:

- **Feedback score**: Total positive feedback count
- **Feedback percentage**: % positive (e.g., 99.2%)
- **Top Rated Seller**: eBay designation for high-performing sellers
- **Seller location**: Domestic vs. international shipping affects buyer pool

Behavioral signals:

- **Number of bids**: More competition typically = higher price (but can't use as feature for prediction at listing time)
- **Number of watchers**: Indicates interest level (some scrapers track this)
- **View count**: Traffic to the listing (may be available through seller hub)

Textual Feature Engineering from eBay

Research shows that **listing presentation quality** significantly affects prices. Bodoh-Creed et al. extracted:

1. **Bag-of-words features**: 190+ keyword indicators from titles/descriptions
    - Condition keywords: "new", "mint", "excellent", "worn", "damaged"
    - Edition signals: "first edition", "signed", "limited", "hardcover"
    - Completeness: "complete", "missing", "dust jacket", "all pages"
2. **Style features**:
    - **Description length**: Character count, word count
    - **HTML formatting**: Presence of bold, italics, bullet points
    - **Uppercase usage**: Proportion of capitals (excessive capitals may signal unprofessionalism)

    - **Number of photos**: More images correlates with higher prices (shows item thoroughly)
    - **Stock photo indicator**: Using publisher's image vs. actual item photo
3. **Dimensionality reduction**:
    - Applied PCA to reduce 190 text features → 70 principal components (98% variance)
    - Used LDA (Latent Dirichlet Allocation) for topic modeling
    - Both approaches prevented overfitting while retaining information

Critical finding: Including these textual features boosted Random Forest R^2 from 0.20 to **0.42**—text more than doubled explanatory power.

Practical lesson: Even simple presence/absence of keywords ("signed", "first edition") can be highly predictive. One study used Naive Bayes on title text alone and outperformed complex algorithms for price range classification.

Behavioral Economics Considerations

eBay auctions introduce **psychological and strategic factors** beyond intrinsic item value:

1. **Winner's curse**: Winners of auctions may have overpaid (highest bidder had highest valuation, possibly inflated)
    - **Implication**: Auction prices might overestimate typical buyer willingness-to-pay
    - **Mitigation**: Weight fixed-price sales more heavily, or model auction premium separately
2. **Auction timing effects**:
    - **Sunday evening** auctions often achieve higher prices (more casual browsers online)
    - **Holiday periods** may have fewer bidders (lower competition) or more gift buyers (higher demand)
    - **Last-minute bidding (sniping)**: Final price emerges in last seconds (hard to predict mid-auction)
3. **Anchoring on starting price**:
    - Low starting price may attract more bidders but risks underselling
    - High starting price may deter participation
    - **Modeling challenge**: Starting price is both a feature and a strategic choice (endogenous)
4. **Shipping psychology**:
    - Buyers mentally separate item price + shipping
    - $10 item + $5 shipping may perform worse than $13 item + $2 shipping (even though total is same)
    - Some sellers game this (low item price, high shipping to reduce eBay fees)

Recommendation for modeling:

- Train separate models for **auction** vs. **Buy It Now** listings (different dynamics)
- Or include **listing_format** as a feature and let the model learn differential pricing
- For a user-facing price suggestion tool, recommend **Buy It Now** pricing (more stable, predictable) unless user specifically wants auction strategy advice

3.3 Amazon Marketplace Data

Amazon's third-party marketplace is the largest used book platform but presents unique data collection challenges.

Data Sources

1. **Amazon Product Advertising API**:
    - Requires approved developer account
    - Returns: Current lowest used price, number of used offers, sales rank
    - **Limitation**: No historical price data or individual seller prices (only current lowest)
2. **Price tracking services**:
    - **CamelCamelCamel**: Tracks new, used, and refurbished prices over time
    - **Keepa**: Similar tracking with charting and alerts
    - These services scrape Amazon continuously and build historical databases
    - **Access**: Some offer APIs or data exports (paid services)
3. **Web scraping**:
    - Amazon's product pages list all current offers (with prices, seller ratings, condition)
    - Can build snapshots of competitive landscape
    - **Challenges**: Anti-scraping measures, IP blocking, captchas, robots.txt compliance
    - **Legal**: Terms of Service generally prohibit automated access without permission
4. **Amazon Seller Central** (for active sellers):
    - Provides detailed sales data for your own listings
    - Marketplace insights (competitive pricing for your inventory)
    - Can't access other sellers' transaction data

Key Features from Amazon

Static book attributes (from Amazon catalog):

- **ASIN/ISBN**: Unique identifier
- **Title, Author, Publisher**: Basic metadata
- **Publication date**: Determines book age
- **List price**: Original retail price (useful baseline)
- **Format**: Hardcover, Paperback, Mass Market, Spiral-bound
- **Number of pages**: Size proxy
- **Dimensions/weight**: Shipping cost factors

Dynamic marketplace data:

- **Lowest used price**: Current minimum across all sellers
- **Number of used offers**: Supply indicator (more offers = more competition = lower price)
- **Amazon sales rank**: Lower rank = higher recent sales velocity (demand proxy)
    - Ranks are categorical by department (e.g., #245 in Books > Textbooks > Engineering)
    - Updated hourly based on sales
- **Prime availability**: Some sellers offer FBA (Fulfilled by Amazon) used books (Prime-eligible)

Social proof / popularity:

- **Average rating**: 1-5 stars
- **Number of ratings**: Total review count (popularity indicator)
- **Reviews**: Text reviews (can extract sentiment, themes)
- **"Frequently bought together"**: Indicates if part of popular bundle/series
- **Best Sellers Rank trajectory**: Rising vs. falling (demand trend)

Amazon-Specific Pricing Dynamics

The Buy Box problem:

- Only one seller "wins" the Buy Box (the default purchase option)
- Buy Box algorithm considers: price, shipping speed, seller rating, fulfillment method
- This creates a **race to the bottom** for commodity items (like textbooks with many sellers)
- **Implication**: Lowest price often dominates, making price prediction easier but leaving less room for premium pricing

Automated repricing:

- Many professional sellers use repricing software that continuously adjusts prices to stay competitive
- Creates rapid price fluctuations (prices can change hourly)
- Can trigger price wars where algorithms respond to each other, driving prices down
- **Modeling challenge**: Snapshot data may not reflect stable equilibrium prices

FBA (Fulfilled by Amazon) premium:

- FBA sellers ship inventory to Amazon warehouses; Amazon handles shipping
- Prime-eligible (free 2-day shipping for Prime members)
- FBA items often command **10-20% price premium** over FBM (merchant-fulfilled)
- **Feature consideration**: Include fulfillment_method if data available

Edition dynamics (especially textbooks):

- New edition release causes **immediate price collapse** for previous editions (often 50-70% drop)
- Used copies of current edition retain 50-80% of new price
- Out-of-print editions of classic texts may **appreciate** (scarcity value)

Combining Amazon and eBay Data

Complementary strengths:

- **Amazon**: Better for current market state, high-volume commodity books (textbooks)
- **eBay**: Better for collectibles, auction dynamics, detailed condition descriptions

Hybrid approach:

1. Train initial model on eBay historical data (abundant, has true sale prices)
2. Augment with Amazon features (sales rank, number of competing offers)
3. Use Amazon's current lowest price as a **competitive baseline** feature
4. Validate on both platforms separately (different user bases, price levels)

Example feature set for a book:

- ISBN: 978-0134685991
- Title: "Effective Java 3rd Edition"
- Author: "Joshua Bloch"
- Format: Paperback
- Pub_Year: 2017
- Condition: Very Good
-
- Amazon_Features:
- - Current_Lowest_Used: $35.99
- - Num_Used_Offers: 47
- - Sales_Rank: 1,250 (in Programming)
- - Avg_Rating: 4.8
- - Num_Ratings: 893
-
- eBay_Features:
- - Recent_Sold_Avg: $32.50 (last 30 days, n=8 sales)
- - Recent_Sold_Median: $33.00
- - Avg_Auction_Price: $30.00 (n=3)

- Avg_BIN_Price: $34.00 (n=5)

A model could learn: "For textbooks, eBay typically sells 5-10% below Amazon's lowest used price. With high ratings and active sales, suggest $33-36 for Very Good condition."

3.4 Additional Data Sources

Specialty Book Marketplaces

AbeBooks (rare/collectible books):

- Focuses on antiquarian, rare, first editions
- Prices can be 10-100× higher than standard used books
- Features: Condition grading for collectibles (Fine, Near Fine, Very Good, etc.), edition points, provenance
- **Data access**: Public listings searchable; no official API; scraping challenging

Alibris (academic/specialty):

- Large selection of out-of-print and academic titles
- Often has unique inventory not on Amazon/eBay
- **Use case**: Pricing rare academic texts or niche subjects

BookFinder, Vialibri (metasearch):

- Aggregate prices across multiple marketplaces
- Useful for competitive pricing research
- Can establish price ranges for rare items with sparse data

Library Sales and Institutional Data

Library Friends sales:

- Public libraries often sell withdrawn books at fixed prices ($1-5 typically)
- Not useful for retail pricing but shows floor prices
- Occasionally rare items mispriced (arbitrage opportunity for resellers)

Textbook buyback programs:

- College bookstores, online buyback services (Chegg, BookFinder)
- Offer wholesale prices (what they'll pay sellers)
- Useful for establishing **minimum viable prices** (if buyback offers $15, retail should be >$20)
- Seasonal patterns: Buyback prices peak before semesters, crash after drop/add period

External Metadata Sources

Goodreads API:

- Average ratings, number of ratings, shelves (to-read counts)
- Reviews (text for sentiment analysis)
- Book series information, genre tags

- Author popularity (number of followers)
- **Strength**: Best source for social/popularity metrics

Google Books API:

- Book descriptions/synopses
- Preview availability (may indicate demand)
- Subject classifications
- Related books (for collaborative filtering approaches)

Open Library / WorldCat:

- Comprehensive bibliographic data
- Library holdings counts (how many libraries own it = proxy for importance/demand)
- Edition relationships (maps different ISBNs for same work)

Publisher websites:

- Official list prices (for comparison)
- In-print status (out-of-print books may have scarcity premium)
- Release dates for upcoming editions (indicates current edition will depreciate)

3.5 Critical Features for Book Pricing

After surveying data sources, we can identify the most predictive features across studies and competitions:

Tier 1: Essential Features (Always include)

1. **Condition** (categorical or ordinal: New > Like New > Very Good > Good > Acceptable)
    - Single most important feature for used items
    - Can shift prices **50-80%** for same book
    - Must be consistently encoded across training data
2. **Edition / Publication Year**
    - For textbooks: Current edition vs. outdated determines viability
    - For collectibles: First edition, first printing adds significant premium
    - Age interaction with genre (novels depreciate faster than classics)
3. **Format** (Hardcover vs. Paperback vs. Mass Market)
    - Hardcovers typically command **20-40%** premium for same title/condition
    - Mass market paperbacks are cheapest format
4. **ISBN / Title / Author**
    - Identifier for joining external data
    - Title text contains signals (keywords like "signed", "limited", "illustrated")
    - Author popularity affects demand
5. **Genre / Category**

- Textbooks vs. Fiction vs. Nonfiction vs. Children's vs. Collectibles behave entirely differently
- Some genres hold value (technical references), others depreciate rapidly (celebrity memoirs)

Tier 2: High-Value Features (Include when available)

6. **Popularity metrics**:
   - **Average rating** (4.5+ stars often commands premium)
   - **Number of ratings** (bestsellers have thousands; obscure books have <10)
   - **Sales rank** on Amazon (lower rank = higher recent sales = better liquidity)
7. **Textual content**:
   - **Synopsis / description**: Genre, themes, topics (using TF-IDF, LDA topics, or BERT embeddings)
   - **Reviews**: Sentiment, praise words ("classic", "essential", "masterpiece" suggest lasting value)
   - **Listing title** (for eBay): Condition mentions, special features
8. **Competitive landscape**:
   - **Number of competing offers** (more supply = lower prices)
   - **Lowest competing price** (market baseline)
   - **Price distribution** of competitors (tight clustering vs. wide spread)
9. **Historical prices**:
   - **Average recent sale price** for this ISBN (must avoid leakage)
   - **Price trend** (rising, stable, falling)
   - **Original list price** (used prices often anchor to this)
10. **Scarcity indicators**:
    - **Out of print** status
    - **Limited edition** flags
    - **Signed copies** (can double or triple value)
    - **First edition, first printing** (for collectibles)

Tier 3: Supplementary Features (Nice to have)

11. **Temporal features**:
    - **Season** (textbook demand spikes August/January)
    - **Days since publication** (age in a more continuous form)
    - **Upcoming movie/TV adaptations** (demand spikes; requires external data)
12. **Physical attributes**:
    - **Number of pages** (size/weight affects shipping, perceived value)
    - **Illustrations / photos** (art books, technical manuals)
    - **Dust jacket present** (for hardcover collectibles)
13. **Platform-specific**:
    - **Fulfillment method** (FBA vs. FBM on Amazon)
    - **Auction vs. BIN** on eBay
    - **Seller rating** (higher-rated sellers can charge slightly more)

14. **Listing quality** (eBay):
    - **Number of photos**
    - **Description length**
    - **HTML formatting quality**

3.6 Feature Engineering: From Raw Data to Model Inputs

Raw features often need transformation to be maximally useful.

Numeric Transformations

Log transforms (for skewed distributions):

python

```
df['log_num_ratings'] = np.log1p(df['num_ratings'])  # log(1 + x) handles zeros
df['log_sales_rank'] = np.log1p(df['sales_rank'])
df['log_pages'] = np.log1p(df['num_pages'])
```

- Compresses high values, spreads low values
- Makes linear models work better (linear in log-space = exponential in original space)

Age calculations:

python

```
current_year = 2025
df['book_age'] = current_year - df['publication_year']
df['age_squared'] = df['book_age'] ** 2  # Allows U-shaped relationships
```

- Linear age may not capture that very old books (antiques) appreciate
- Polynomial terms let model learn curves

Binning (discretizing continuous features):

python

```
df['age_category'] = pd.cut(df['book_age'],
                            bins=[0, 1, 5, 10, 50, 150],
                            labels=['New Release', 'Recent', 'Backlist', 'Older', 'Vintage'])
```

- Useful when relationship is step-wise rather than smooth
- Tree models don't need this, but can help linear models

Normalization (for neural networks):

python

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[['rating', 'num_ratings', 'book_age']] = scaler.fit_transform(
    df[['rating', 'num_ratings', 'book_age']]
)
```

- Makes features comparable in scale
- Prevents large-magnitude features from dominating

Categorical Encoding

One-hot encoding (for low-cardinality categoricals):

python

```
pd.get_dummies(df['format'])  # Creates Format_Hardcover, Format_Paperback, etc.
```

- Standard for genres, format, condition (if treating as nominal)
- Can explode dimensionality if many categories

Ordinal encoding (for ranked categories):

python

```
condition_map = {'Acceptable': 1, 'Good': 2, 'Very Good': 3, 'Like New': 4, 'New': 5}
df['condition_ordinal'] = df['condition'].map(condition_map)
```

- Preserves order (Better condition → higher numeric value)
- Assumes linear spacing (may not be true: New to Like New is smaller gap than Good to Acceptable)

Target encoding (for high-cardinality categoricals):

python

```
# For each author, compute average price from training data
author_mean_price = train.groupby('author')['price'].mean()
df['author_target_encoded'] = df['author'].map(author_mean_price)
```