Advanced Topics - Feature Engineering

Feature Selection, Creation, and Transformation Techniques

Sarwan Ali

Department of Computer Science
Georgia State University

Crafting Better Features

Today's Learning Journey

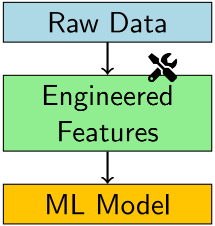
What is Feature Engineering?

Feature Engineering is the process of:

- Selecting relevant features
- Creating new features from existing ones
- Transforming features for better performance

Key Insight

"Feature engineering is often the difference between a good model and a great model"



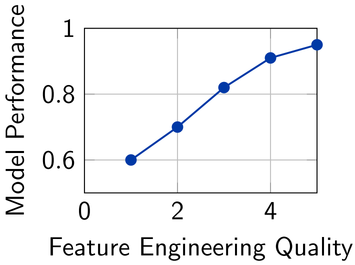
Why Feature Engineering Matters

Impact on Model Performance:

- Improves accuracy and generalization
- Reduces overfitting
- Speeds up training
- Makes models more interpretable

Real-World Importance:

- 80% of ML project time
- Domain expertise crucial
- Often more impactful than algorithm choice



Feature Selection Overview

Definition

Feature selection is the process of selecting a subset of relevant features for model construction.

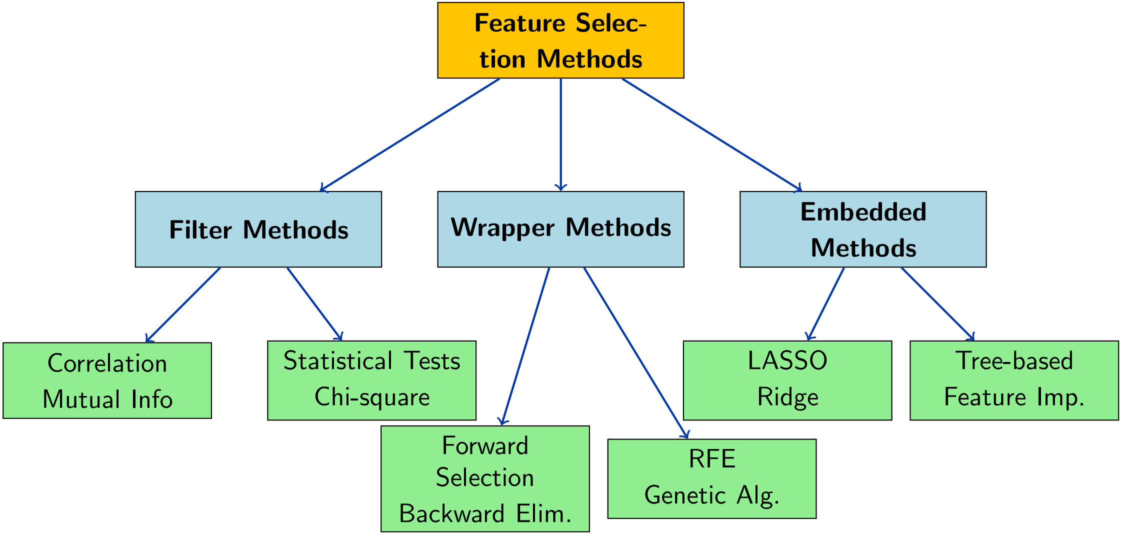
Benefits:

- Reduces overfitting
- Improves accuracy
- Reduces training time
- Simplifies model interpretation
- Reduces storage requirements

Challenges:

- Curse of dimensionality
- Feature interactions
- Computational complexity
- Domain knowledge required

Types of Feature Selection



Filter Methods

Characteristics:

- Independent of ML algorithm
- Fast and scalable
- Based on statistical properties

Correlation-based:

- Pearson correlation
- Spearman correlation
- Kendall's tau

Information-based:

- Mutual Information
- Information Gain
- Gain Ratio

Statistical Tests:

- Chi-square test
- ANOVA F-test
- t-test

Variance-based:

- Variance threshold
- Quasi-constant features

Formula: Mutual Information

$$I(X;Y) = \sum_{x\in X} \sum_{y\in Y} p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

Wrapper Methods

Characteristics:

- Use ML algorithm performance as criterion
- More accurate but computationally expensive
- Risk of overfitting

Forward Selection:

1. Start with empty set
2. Add best feature iteratively
3. Stop when no improvement

Backward Elimination:

1. Start with all features
2. Remove worst feature iteratively
3. Stop when performance degrades

Recursive Feature Elimination (RFE):

1. Train model with all features
2. Rank features by importance
3. Remove least important
4. Repeat until desired number

Pros & Cons

Pros: Considers feature interactions
Cons: Computationally expensive

Embedded Methods

Characteristics:

- Feature selection integrated into model training
- Balance between filter and wrapper methods
- Algorithm-specific

Regularization-based:

- LASSO (L1 regularization)
- Ridge (L2 regularization)
- Elastic Net

Tree-based Methods:

- Random Forest feature importance
- Gradient Boosting feature importance
- Permutation importance

LASSO Objective

$\min_\beta \frac{1}{2n}||y - X\beta||_2^2 + \lambda||\beta||_1$

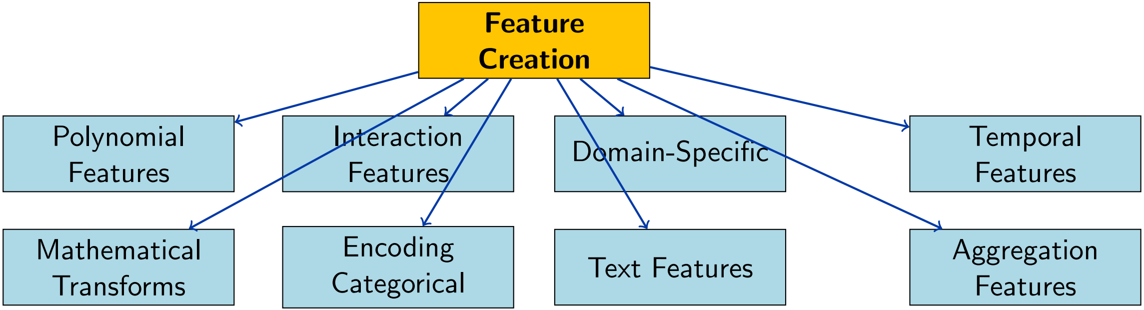
Feature Importance

Based on how much each feature decreases impurity when used for splits

Feature Creation Techniques

Definition

Feature creation involves generating new features from existing ones to capture hidden patterns and relationships.



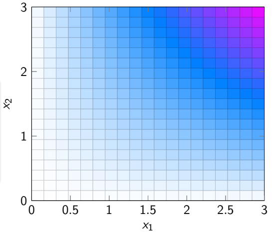
Polynomial and Interaction Features

Polynomial Features:

- Capture non-linear relationships
- Powers of existing features
- Example: $x, x^2, x^3, \dots$

Interaction Features:

- Capture feature relationships
- Products of feature pairs
- Example: $x_1 \times x_2$



Mathematical Form

For features x_1, x_2:
$\{1, x_1, x_2, x_1^2, x_1x_2, x_2^2, \dots\}$

Example: House Prices

Original: size, bedrooms
Polynomial: size^2, bedrooms^2
Interaction: size $\times$ bedrooms

Mathematical Transformations

Log Transformations:

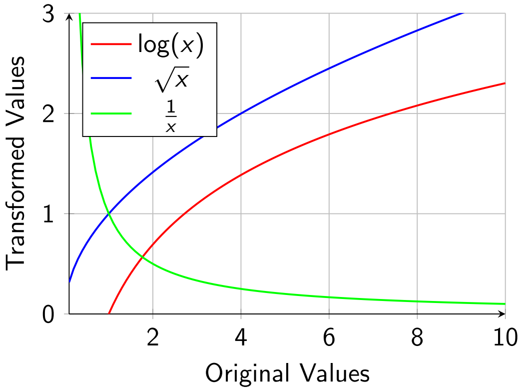
- Handle skewed distributions
- $\log(x+1)$ for zero values
- Makes data more normal

Square Root:

- Moderate skewness reduction
- $\sqrt{x}$ or $\sqrt{x+c}$
- Preserves zero values

Reciprocal:

- $1/x$ transformation
- Changes scale dramatically
- Careful with zero values



Box-Cox Transformation

$$y(\lambda) = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(x) & \text{if } \lambda = 0 \end{cases}$$

Encoding Categorical Variables

One-Hot Encoding:

- Binary columns for each category
- Suitable for nominal data
- Can create many columns

Label Encoding:

- Integer mapping of categories
- Suitable for ordinal data
- Implies ordering

Target Encoding:

- Mean target value per category
- Risk of overfitting
- Useful for high cardinality

Color	Red	Blue	Green
Red	1	0	0
Blue	0	1	0
Green	0	0	1
Blue	0	1	0

Table: One-Hot Encoding

Advanced Techniques:

- Binary encoding
- Frequency encoding
- Hash encoding
- Embedding (for deep learning)

Temporal Feature Engineering

Time-based Features:

- Hour, day, month, year
- Day of week, weekend indicator
- Season, quarter
- Business hours indicator

Cyclical Encoding:

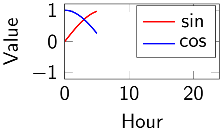
- Sine/cosine transformations
- Preserve cyclical nature
- Example: hour of day

Lag Features:

- Previous time period values
- Moving averages
- Rolling statistics

Date Differences:

- Days since last event
- Time to next holiday
- Age calculations



Cyclical Encoding

$\sin\left(\frac{2\pi\cdot\text{hour}}{24}\right)$
$\cos\left(\frac{2\pi\cdot\text{hour}}{24}\right)$

Feature Scaling and Normalization

Why Scale Features?

Different features have different scales, which can bias algorithms that use distance measures.

Min-Max Scaling:

- Scales to [0,1] range
- Preserves relationships
- Sensitive to outliers

Formula

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Standardization (Z-score):

- Mean = 0, Std = 1
- Assumes normal distribution
- Not bounded to specific range

Formula

$$x_{std} = \frac{x - \mu}{\sigma}$$

Robust Scaling:

- Uses median & IQR, Less sensitive to outliers

Formula

$$x_{robust} = \frac{x - median(x)}{IQR(x)}$$

Unit Vector Scaling:

- Scales to unit norm, Useful for text data

Formula

$$x_{unit} = \frac{x}{||x||_2}$$

Handling Skewed Distributions

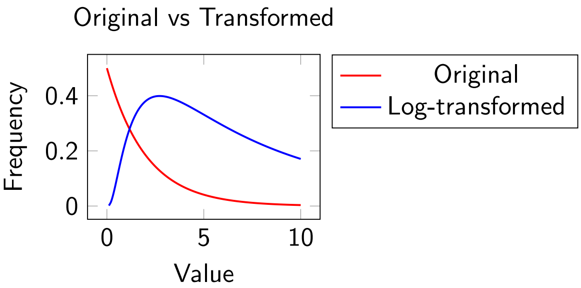
Identifying Skewness:

- Skewness coefficient
- Visual inspection (histograms)
- Q-Q plots

Skewness Formula

$$S = \frac{E[(X-\mu)^3]}{\sigma^3}$$

$S > 1$: Highly right-skewed
$|S| < 0.5$: Approximately normal



When to Transform:

- Linear models assume normality
- Improve model performance
- Better visualization

Dimensionality Reduction

Principal Component Analysis (PCA):

- Linear transformation to uncorrelated components
- Maximizes variance in lower dimensions
- Useful for visualization and noise reduction

PCA Steps:

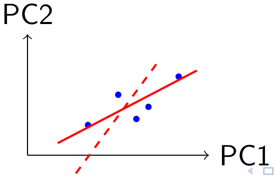
1. Standardize the data
2. Compute covariance matrix
3. Find eigenvalues/eigenvectors
4. Select principal components
5. Transform data

Variance Explained

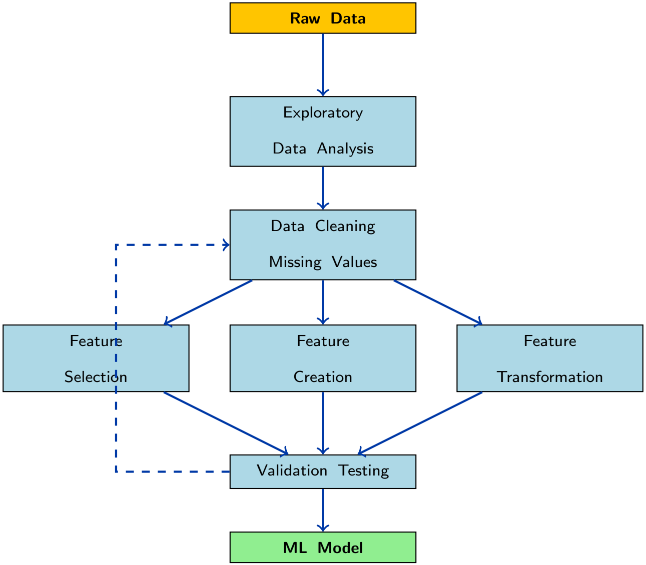
$$\text{Ratio} = \frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j}$$

Other Techniques:

- t-SNE (non-linear)
- UMAP (preserves structure)
- Linear Discriminant Analysis
- Independent Component Analysis



Feature Engineering Pipeline



Best Practices

Domain Knowledge:

- Understand business context
- Collaborate with domain experts
- Research existing literature
- Consider physical constraints

Data Understanding:

- Explore data distributions
- Identify missing patterns
- Check for data leakage
- Understand temporal aspects

Validation Strategy:

- Use proper cross-validation
- Avoid look-ahead bias
- Test on holdout set
- Monitor for overfitting

Iterative Approach:

- Start simple, add complexity
- Document all transformations
- Version control features
- A/B test feature changes

Golden Rule

Always validate features on unseen data before deploying to production!

Common Pitfalls and How to Avoid Them

Data Leakage:

- Using future information
- Target leakage
- Wrong: Include post-event features
- Right: Only use historical data

Inconsistent Preprocessing:

- Different train/test preprocessing
- Data scaling after splitting
- Wrong: Scale entire dataset
- Right: Fit on train, transform test

Overfitting Features:

- Too many engineered features
- Complex interactions on small data
- Wrong: 1000 features, 100 samples
- Right: Use regularization

Ignoring Feature Interactions:

- Missing important combinations
- Not considering non-linearity
- Wrong: Only linear features
- Right: Test interactions systematically

Case Study: Predicting House Prices

Dataset: House characteristics and sale prices

Original Features:

- Size (sq ft)
- Bedrooms, Bathrooms
- Age of house
- Neighborhood
- Lot size

Engineered Features:

- Size per bedroom ratio
- Age categories (new/old)
- Price per sq ft (for similar houses)
- Distance to amenities
- Seasonal indicators

Feature Selection:

- Remove highly correlated features
- Use LASSO for automatic selection
- Keep features with importance > 0.05

Transformations:

- Log(price) - target variable
- StandardScaler for continuous
- One-hot encode neighborhoods

Results

Baseline (raw features): RMSE = \$45,000
After feature engineering: RMSE = \$32,000 (29% improvement)

Case Study: Text Classification

Problem: Classify customer reviews as positive/negative

Text Preprocessing:

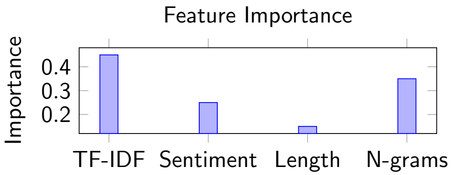
- Lowercase conversion
- Remove punctuation/numbers
- Stop word removal
- Stemming/Lemmatization

Feature Creation:

- TF-IDF vectors
- N-gram features (1-3)
- Sentiment scores
- Text length metrics

Advanced Features:

- Word embeddings (Word2Vec)
- Part-of-speech tags
- Named entity counts
- Readability scores



Performance

Bag of Words: Accuracy = 82%
Engineered Features: Accuracy = 89% (+7%)

Tools and Libraries

Python Libraries:

- scikit-learn: Feature selection, scaling
- pandas: Data manipulation
- numpy: Mathematical operations
- feature-engine: Specialized FE
- category_encoders: Categorical encoding

Automated FE:

- featuretools: Automated feature generation
- tsfresh: Time series features
- boruta: Feature selection

R Libraries:

- caret: Comprehensive ML toolkit
- recipes: Feature engineering recipes
- VIM: Missing value imputation

Specialized Tools:

- H2O AutoML: Automated feature engineering
- DataRobot: Enterprise AutoML
- RAPIDS: GPU-accelerated FE

Code Example

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_train)
```

Evaluation Metrics for Feature Engineering

Model Performance:

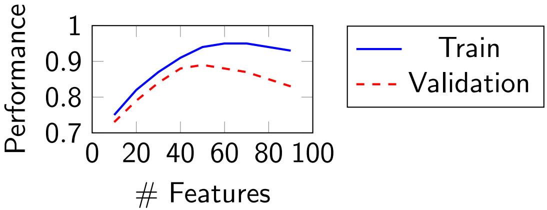
- Cross-validation scores
- Holdout test performance
- Learning curves
- Bias-variance analysis

Feature Quality:

- Feature importance scores
- Correlation with target
- Stability across time
- Business interpretability

Computational Metrics:

- Training time
- Inference speed
- Memory usage
- Storage requirements



Feature Engineering Success Metrics

Good FE: Improves validation score, maintains interpretability
Overfitting: Train score increases, validation score decreases

Advanced Topics Preview

Automated Feature Engineering:

- Deep Feature Synthesis
- Genetic Programming
- Neural Architecture Search
- AutoML platforms

Deep Learning Features:

- Learned embeddings
- Representation learning
- Transfer learning features
- Attention mechanisms

Domain-Specific FE:

- Image: HOG, SIFT, CNN features
- Audio: MFCC, spectrograms
- Time Series: Fourier transforms
- Graph: Node embeddings

Real-time FE:

- Streaming feature computation
- Online learning features
- Feature stores
- Edge computing features

Future Directions

Feature engineering is evolving toward automated, domain-aware, and real-time systems

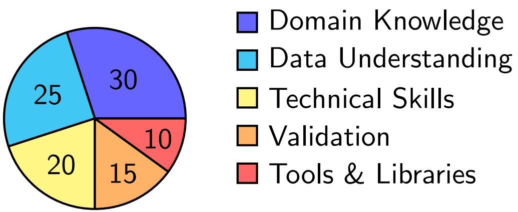
Summary

Key Takeaways:

- Feature engineering is crucial for ML success
- Combines domain knowledge with data science
- Iterative process requiring validation
- Balance complexity with interpretability

Remember:

- Start simple, add complexity gradually
- Always validate on unseen data
- Document your feature engineering pipeline
- Consider computational constraints



Success Formula

Next Steps

Practice with real datasets, experiment with different techniques, and always measure the impact!

Questions & Discussion

Think about:

- What features might be important in your domain?
- How would you handle missing values?
- What transformations make sense for your data?
- How would you validate your feature engineering?

sali85@student.gsu.edu