Feature Selection and Engineering

Outline

Introduction

Introduction

What is Feature Engineering?

Feature Engineering is the process of using domain knowledge to extract features from raw data that make machine learning algorithms work better.

What is Feature Selection?

Feature Selection is the process of selecting a subset of relevant features for use in model construction. It helps reduce overfitting, improves accuracy, and decreases training time.

Feature Engineering Techniques

Handling Missing Values

Replacing or flagging missing data to ensure model integrity.

Examples:

- **Imputation**: Fill missing values using Mean/Median/Mode.
- **Interpolation**: Estimate missing values based on nearby data points.
- **Model-based**: Use KNN or MICE to predict missing values.
- **Flag Missingness**: Create a binary indicator variable for missing data.

Encoding Categorical Variables

Convert categorical data into numerical format for ML models.

Examples:

- Label Encoding (Ordinal data)
- One-Hot Encoding (Nominal data)
- Target Encoding (Category replaced with target mean)
- Frequency Encoding (Category replaced with frequency count)

Scaling and Normalization

Adjust feature values to improve model efficiency and stability.

Examples:

- Min-Max Scaling (Rescale values between 0 and 1)
- Standardization (Z-score normalization)
- Robust Scaling (Handles outliers effectively)

Date and Time Features

Extract meaningful information from timestamps.

Examples:

- Extract year, month, weekday, hour
- Compute duration (e.g., time since last purchase)

Text Feature Engineering

Convert text into numerical representations for ML models.

Examples:

- TF-IDF, Bag of Words
- N-grams for sequence analysis
- Word Embeddings (Word2Vec, GloVe, BERT)

Polynomial and Interaction Features

Enhance feature complexity for non-linear relationships.

Examples:

- Use PolynomialFeatures from scikit-learn
- Multiply or add features for richer relationships

Aggregation and GroupBy Features

Summarize data points based on logical groups.

Examples:

- Compute average purchase per user
- Count-based features (e.g., number of transactions)

Domain-Specific Features

Custom features designed from industry knowledge.

Examples:

- BMI calculated from weight and height
- Risk scores in fraud detection

Feature Selection Techniques

Filter Methods

Feature selection based on statistical measures independent of the model.

Examples:

- **Correlation Thresholding**: Remove highly correlated features (e.g., remove one of "height" and "weight" if correlation ¿ 0.9).
- **Chi-Square Test**: Assess categorical features (e.g., selecting significant features for predicting customer churn).
- **ANOVA F-test**: Evaluate numerical features against categorical targets (e.g., testing if different income levels affect purchase decisions).
- **Mutual Information**: Measure dependence between feature and target (e.g., selecting words that strongly predict spam emails).

Wrapper Methods

Feature selection performed by iteratively testing subsets using a model.

Examples:

- **Recursive Feature Elimination (RFE)**: Remove least important features step-by-step (e.g., selecting the top 10 features for predicting house prices).
- **Sequential Feature Selection (SFS)**: Add/remove features based on model accuracy (e.g., improving a fraud detection model by iteratively selecting important transaction features).
- **Forward Selection**: Start with no features and add one at a time (e.g., selecting the best combination of factors affecting customer retention).

Embedded Methods

Feature selection performed automatically during model training.

Examples:

- **Lasso Regression (L1 Regularization)**: Shrinks less important feature weights to zero (e.g., selecting the most relevant features for predicting house prices).
- **Ridge Regression (L2 Regularization)**: Reduces magnitude but does not eliminate features (e.g., ensuring smooth feature importance ranking in medical diagnoses).
- **Tree-based Feature Importance**: Rank features using models like Random Forest or XGBoost (e.g., identifying the top predictors for customer churn).

Curse of Dimensionality

Curse of Dimensionality: As the number of features increases, the volume of the feature space increases exponentially, making it harder to find patterns.

Dimensionality Reduction

Dimensionality Reduction Techniques *Reduce feature space while preserving the most relevant information.*

Examples:

- **Principal Component Analysis (PCA)**: Transform correlated features into independent components (e.g., reducing hundreds of financial metrics into principal components for stock price forecasting).
- **Linear Discriminant Analysis (LDA)**: Maximizes class separability while reducing dimensions (e.g., distinguishing between cancer vs. non-cancer patients based on genetic markers).
- **t-SNE / UMAP**: Useful for data visualization (e.g., clustering customers based on behavioral similarities).

Looks Good on Paper: Why Your Model Fails in the Real World

Data Leakage

Definition: When information from outside the training dataset leaks into the model training process. **"The Devil in Disguise"**

- **Target Leakage:** Using labels or future data in training
- **Train-Test Contamination:** Preprocessing before data split
- **Temporal Leakage:** Using future time data to predict the present

Prevention:

- Split data before any preprocessing
- Avoid features unavailable at inference
- Use time-aware splitting for time series

Data Augmentation

Purpose: Increase dataset size/diversity to improve generalization.

Common Techniques

- **Images:** Flip, rotate, crop, noise
- **Text:** Synonym replacement, back-translation
- **Time Series:** Jittering, slicing

Cautions:

- Don't change labels via augmentation
- Apply only to training data

Common Pitfalls in ML

Pitfall	Solution
Overfitting	Regularization, cross-validation
Underfitting	More features or complex model
Imbalanced data	SMOTE, class weights
Data leakage	Proper split before processing
Incorrect preprocessing	Standardize after splitting

Cross-Validation

Goal: Evaluate model stability using resampling.**"On Training set only"**

Types of Cross-Validation

- **K-Fold**: Generic, good default
- **Stratified K-Fold**: Maintains class balance
- **TimeSeriesSplit**: For time-dependent data
- **Group K-Fold**: Prevents group overlap

Tips:

- Use nested CV for model + hyperparameter selection
- Always validate on **unseen data**

Summary

- Data leakage can ruin model performance — always split first!
- Use augmentation wisely — only on training data
- Avoid common ML mistakes: overfitting, imbalance, poor validation
- Cross-validation is key for robust evaluation

Practice carefully, validate robustly, deploy confidently!

Thank you!