Machine Learning

Dr.Hajialiasgari

Tehran University
Of
Medical Science

December 8, 2024

TEHRAN UNIVERSITY
OF
MEDICAL SCIENCES

- Feature engineering is crucial in machine learning as it transforms raw data into meaningful features, enhancing model performance and accuracy. By creating, selecting, and preprocessing features, it helps algorithms better understand patterns, leading to improved predictions and insights.

- `Relevant:` Directly related to the target variable and contributes to prediction accuracy.
- `Independent:` Minimally correlated with other features to avoid redundancy.
- `Discriminative:` Distinguishes between different classes or outcomes effectively.
- `Robust:` Handles noise, missing values, and outliers without degrading model performance.

- `Numerical Features:` Quantitative data that represent measurable quantities (e.g., age, salary).
- `Categorical Features:` Qualitative data with discrete values representing categories (e.g., gender, city).
- `Ordinal Features:` Categorical data with a meaningful order or ranking (e.g., education level: high school, bachelor's, master's).
- `Binary Features:` Variables with only two possible values (e.g., 0/1, true/false).

- `Textual Features:` Data in text format requiring techniques like tokenization or embedding (e.g., customer reviews).
- `Temporal Features:` Data involving time, such as timestamps or durations (e.g., transaction date).
- `Spatial Features:` Data related to location or geography (e.g., latitude and longitude).
- `Derived Features:` Features created from raw data using transformations, combinations, or domain knowledge.

Categorical Features

- `Imputation with a Constant:` Replace missing values with a placeholder such as `'Unknown'` or `'Missing'`. Useful when the absence of data has its own meaning.
- `Mode Imputation:` Replace missing values with the most frequent category. Suitable for features with a clear dominant class.
- `Imputation Based on Other Features:` Predict missing values using other related features. Requires advanced techniques like regression or classification models.
- `Frequency Encoding:` Replace missing values with the frequency or probability of each category.

Categorical Features

- `Custom Imputation:` Use domain knowledge to assign meaningful values. Effective when the missingness has a known context.
- `Separate Category:` Treat missing values as a separate category. Ideal for algorithms that can handle additional classes, such as decision trees.
- `Remove Rows/Columns:` Remove data points or features with too many missing values. Only appropriate when the missing data is non-critical or minimal.

Numerical Features

- `Mean Imputation:` Replace missing values with the mean of the feature. Works well for data with a normal distribution.
- `Median Imputation:` Replace missing values with the median of the feature. Suitable for skewed data or features with outliers.
- `Mode Imputation:` Replace missing values with the mode (most frequent value). Useful when a single value dominates the feature.
- `Imputation Using Other Features:` Predict missing values using related features through regression or machine learning models. Effective for datasets with strong feature relationships.

Numerical Features

- `Interpolation:` Estimate missing values using trends in the data (e.g., linear or polynomial interpolation). Works well for time series or sequential data.
- `Filling with a Constant:` Replace missing values with a specific constant, such as 0. Appropriate when the missing values represent an absence.
- `Remove Rows/Columns:` Drop rows or features with too many missing values. Suitable when the proportion of missing values is high and the feature is non-critical.
- `Use KNN Imputation:` Fill missing values by averaging the values of the `k`-nearest neighbors. Effective when similar data points are present.

- These features are especially beneficial in time-series forecasting, sales analysis, and any context where time plays a significant role in influencing patterns or behaviors.

- `Day of the Week:` Indicates the specific day (e.g., Monday, Tuesday) to capture weekday or weekend effects.
- `Month:` Represents the month (e.g., January, February) to account for seasonal variations.
- `Year:` Useful for capturing long-term trends or changes over years.
- `Quarter:` Denotes the quarter of the year (e.g., Q1, Q2) to capture business cycles or seasonal trends.
- `Day of the Month:` Specifies the day within a month (e.g., 1st, 15th).

- `Week of the Year:` Captures the week number in a year (e.g., Week 1, Week 52).
- `Is Weekend/Weekday:` A binary feature indicating whether a date falls on a weekend or a weekday.
- `Is Holiday:` Indicates whether the date is a public or special holiday, often determined based on local calendars.
- `Season:` Classifies the date into a season (e.g., Spring, Summer) to capture climate or activity-based trends.
- `Elapsed Time:` Measures the time difference between the given date and a reference date (e.g., days since a start date).

- Feature Synthesis in machine learning refers to the process of creating new features by combining or transforming existing ones.
- These synthesized features can capture complex relationships or patterns that may not be explicitly represented in the raw data, thereby improving model performance. It is a part of feature engineering and often requires domain knowledge or automated tools.

Key Techniques for Feature Synthesis

- `Mathematical Transformations:` Apply arithmetic operations to existing features. *Example:* Combine `length` and `width` to calculate `area`.
- `Polynomial Features:` Generate higher-order combinations of numerical features. *Example:* Squaring or cubing features for non-linear modeling.
- `Aggregation:` Summarize data across groups using operations like mean, median, or sum. *Example:* Average transaction value per customer.
- `Interaction Features:` Combine features to represent their interaction. *Example:* Combine `age` and `income` to analyze purchasing behavior.

Key Techniques for Feature Synthesis

- `Date/Time-Based Features:` Extract insights from timestamps. *Example:* Day of the week, quarter, or duration of an event.
- `Encoding Categorical Features:` Convert categories into meaningful numerical representations. *Example:* Frequency or target encoding.
- `Text Features:` Extract features from text data using NLP techniques. *Example:* TF-IDF or sentiment analysis.
- `Domain-Specific Features:` Leverage domain knowledge to create relevant features. *Example:* Calculate BMI in healthcare using weight and height.

Definition

- Feature scaling is a preprocessing technique used to normalize or standardize the range of independent features (input variables) in a dataset. It ensures that features contribute equally to the model's performance, avoiding bias toward features with larger scales.

Why is Feature Scaling Important?

- `Improves Model Performance:` Many machine learning algorithms, such as gradient descent-based models, perform better when features are scaled consistently.
- `Handles Different Ranges:` Features with vastly different ranges (e.g., income in dollars vs. age in years) can skew the results.
- `Enhances Convergence:` Models like neural networks and SVMs converge faster with scaled features.
- `Reduces Sensitivity to Scale:` Distance-based models (e.g., KNN, K-means) rely on scaled data to calculate accurate distances.

Common Techniques for Feature Scaling

- `Min-Max Scaling (Normalization):` Transforms features to a fixed range, typically [0, 1].
  - **Formula:**

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

  - **Use Case:** Suitable when the data distribution is not Gaussian.
- `Standardization (Z-Score Scaling):` Centers the data around 0 with a standard deviation of 1.
  - **Formula:**

$$x' = \frac{x - \mu}{\sigma}$$

  - **Use Case:** Effective for models requiring Gaussian-like data.

Definition

- Data leakage occurs when information from outside the training dataset is inappropriately used to create a machine learning model, leading to overly optimistic performance metrics during training and poor performance on unseen data.

Why is Data Leakage a Problem?

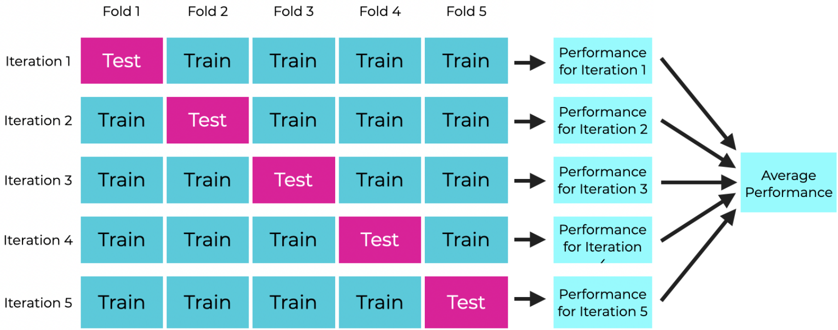
- `Unrealistic Performance Metrics:` The model appears to perform well during training or validation but fails in real-world scenarios.
- `Overfitting:` The model relies on leaked information rather than learning true patterns from the data.
- `Loss of Generalization:` The model cannot generalize to new, unseen data, rendering it ineffective in production.

Common Causes of Data Leakage

- `Target Leakage:` When features contain information about the target that wouldnt be available during prediction. **Example:** Using future sales data to predict current sales.
- `Train-Test Contamination:` Test data is used during training, compromising evaluation. **Example:** Normalizing the entire dataset before splitting.
- `Temporal Leakage:` Using future data to predict past or present outcomes. **Example:** Including post-event data for event prediction.
- `Improper Cross-Validation:` Data from test folds leaks into training folds during validation.

Cross-Validation

CROSS VALIDATION, EXPLAINED



How to Prevent Data Leakage?

- `Careful Feature Selection:` Ensure features do not contain information about the target variable.
- `Correct Data Splitting:` Split the dataset into train, validation, and test sets before any preprocessing.
- `Time-Aware Splitting:` For time-series data, split based on time to avoid future data influencing predictions.
- `Isolation of Test Data:` Keep the test set separate and untouched until the final evaluation.
- `Use of Pipelines:` Automate preprocessing and training to ensure consistent operations across splits.

Definition

- The curse of dimensionality refers to the challenges and inefficiencies that arise when analyzing and organizing data in high-dimensional spaces. As the number of features (dimensions) increases, the amount of data needed to maintain the same level of performance grows exponentially.

Why is it a Problem?

- `Sparse Data in High Dimensions:` As dimensions increase, data points become more spread out, making it difficult for models to find meaningful patterns.
- `Increased Computational Cost:` High-dimensional data requires more memory and computation, slowing down training and prediction.
- `Overfitting Risk:` Models may capture noise instead of meaningful patterns due to the vast feature space.
- `Reduced Model Interpretability:` High-dimensional data makes it harder to understand relationships between features and the target variable.

How to Mitigate the Curse of Dimensionality ?

- `Feature Selection:` Select only the most relevant features for your model.
- `Dimensionality Reduction:` Use techniques like PCA or t-SNE to reduce the number of dimensions.
- `Regularization Techniques:` Prevent overfitting by adding penalties to complex models.
- `Increase Data Volume:` Gather more data to compensate for the increased feature space.

Why is Feature Selection Important?

- `Improves Model Performance:` Removes irrelevant features, allowing the model to focus on the most important information, leading to better accuracy.
- `Reduces Overfitting:` Fewer features reduce the risk of the model capturing noise, preventing overfitting.
- `Enhances Generalization:` A simpler model generalizes better to new, unseen data.
- `Reduces Computational Cost:` With fewer features, the model requires less memory and computation, speeding up training.
- `Improves Interpretability:` A model with fewer features is easier to understand and interpret, which aids in decision-making.

Common Techniques for Feature Selection

- `Correlation Matrix:` Identify highly correlated features and remove the redundant ones.
- `Principal Component Analysis (PCA):` Reduce the dimensionality of the dataset while retaining as much variance as possible by transforming features into principal components.
- `Mutual Information:` Measures the dependency between features and the target variable.
- `Recursive Feature Elimination (RFE):` Iteratively removes features and builds models to identify the most important ones.

For more information and code check the related notebook

End of Feature Engineering