Lecture 11, MTH 400, Fall 2024

Transforming Data, Feature Engineering

R provides several methods and packages for data cleaning and reformatting. Here are some commonly used methods and packages:

1. dplyr: The dplyr package, part of the tidyverse, offers a set of functions for efficient data manipulation. It provides easy-to-use functions like filter(), select(), mutate(), and arrange() for subsetting, selecting columns, creating new variables, and reordering rows in your data frame. dplyr also supports grouping and summarizing data using functions like group_by() and summarize().
2. tidyr: The tidyr package, also part of the tidyverse, focuses on data reshaping and tidying. It provides functions like gather(), spread(), separate(), and unite() to transform data between wide and long formats, split and combine columns, and handle missing values.
3. stringr: The stringr package provides functions for working with strings in R. It offers functions like str_replace(), str_detect(), str_extract(), and str_split() for pattern matching, string substitution, extraction, and splitting. stringr is particularly useful for cleaning and manipulating text data.
4. lubridate: The lubridate package simplifies working with dates and times in R. It provides functions like ymd(), mdy(), hms(), and interval() for parsing and formatting date-time data, extracting components, and performing calculations. lubridate can help with data cleaning tasks involving date and time variables.
5. janitor: The janitor package offers functions for data cleaning and tabular data formatting. It includes functions like clean_names(), remove_empty(), remove_constant(), and tabyl() for cleaning column names, removing empty rows/columns, identifying constant columns, and creating frequency tables.
6. data.table: The data.table package provides enhanced data manipulation capabilities, especially for large datasets. It offers fast and memory-efficient functions like data.table(), setkey(), and := for subsetting, joining, and updating data tables. data.table is known for its speed and efficiency when working with large datasets.
7. Hmisc: The Hmisc package provides a collection of functions for data manipulation and summarization. It includes functions like impute(), describe(), and recode() for imputing missing values, generating descriptive statistics, and recoding variables.

These are just a few examples of methods and packages available in R for data cleaning and reformatting. Depending on your specific needs and the nature of your data, there may be other packages and techniques that can be applied. It's always recommended to refer to the documentation and examples provided by each package for detailed usage instructions.

Feature engineering is the process of transforming raw data into a set of relevant features that can be used to build machine learning models. It involves selecting, creating, and transforming variables in a way that captures the underlying patterns and relationships in the data, making them more suitable for model training. Here's a step-by-step overview of the feature engineering process:

1. Data Understanding: Gain a deep understanding of the data you are working with. Explore the data's characteristics, such as its structure, types of variables, and their meanings. Understand the domain knowledge and the problem you are trying to solve.
2. Data Cleaning: Clean the data by handling missing values, outliers, and inconsistencies. This may involve imputing missing values, removing or correcting outliers, and addressing data quality

issues. Cleaning the data ensures that the subsequent feature engineering steps are based on reliable and consistent data.

3. Feature Selection: Select a subset of relevant features from the available variables. Consider domain knowledge and statistical techniques to identify features that are likely to have a strong relationship with the target variable. This helps reduce dimensionality and focus on the most informative features, which can improve model performance and reduce overfitting.
4. Feature Creation: Create new features based on existing variables or domain knowledge. This can involve combining existing features, performing mathematical operations, deriving statistical measures, or extracting useful information from text, time, or geographical data. Feature creation helps to capture additional patterns and relationships that may not be evident in the original variables.
5. Feature Transformation: Transform features to make them more suitable for model training. This can involve scaling numerical features, encoding categorical variables, or applying mathematical transformations such as logarithmic or power transformations. Feature transformation helps in normalizing the data, reducing the impact of different scales, and improving the model's ability to capture patterns.
6. Feature Encoding: Encode categorical variables into numerical representations that machine learning algorithms can understand. Common encoding techniques include one-hot encoding, label encoding, or target encoding. This ensures that categorical variables are properly represented in a format that can be used for model training.
7. Feature Scaling: Scale numerical features to a similar range to avoid any bias towards variables with larger values. Common scaling techniques include standardization (mean centering and scaling to unit variance) or normalization (scaling to a specific range or distribution).
8. Feature Interactions: Create interaction terms or combinations of features to capture non-linear relationships or interactions between variables. This can involve multiplying, dividing, or applying mathematical operations to pairs or groups of features. Feature interactions can enhance the model's ability to capture complex relationships and improve predictive performance.
9. Iterative Process: Feature engineering is often an iterative process. As you build and evaluate models, you may discover the need for further feature engineering steps or adjustments. This involves analyzing the model's performance, identifying areas for improvement, and refining the features accordingly.
10. Evaluation: Evaluate the engineered features by training machine learning models on the transformed data. Use appropriate evaluation metrics to assess the models' performance and iteratively refine the feature engineering process if needed.

Feature engineering is a crucial step in building effective machine learning models. It requires a combination of domain knowledge, creativity, and data exploration skills to extract meaningful information from raw data and create features that capture the underlying patterns and relationships.

Resources:

1. https://www.geeksforgeeks.org/feature-engineering-in-r-programming/
2. https://r4ds.had.co.nz/transform.html
3. https://www.geeksforgeeks.org/how-to-transform-data-in-r/
4. https://cran.r-project.org/web/packages/dlookr/vignettes/transformation.html
5. https://www.datanovia.com/en/lessons/transform-data-to-normal-distribution-in-r/
6. https://bookdown.org/jaf005/Data-Analysis-with-R/data-transformation.html
7. https://forum.posit.co/t/data-transformation-and-manipulation-in-r-for-plotting/127307
8. https://www.statology.org/transform-data-in-r/

9. https://www.tmwr.org/recipes
10. https://jtr13.github.io/cc20/data-preprocessing-and-feature-engineering-in-r.html
11. https://medium.com/@jscvcds/a-comprehensive-guide-to-feature-engineering-and-selection-in-r-a544e2d1a9ff
12. https://www.hackerearth.com/practice/machine-learning/advanced-techniques/text-mining-feature-engineering-r/tutorial/
13. https://cran.r-project.org/web/packages/finnts/vignettes/feature-engineering.html