



★ Member-only story

Feature Engineering for Data Science Beginners



Ritesh Gupta · Follow

Published in Artificial Intelligence in Plain English · 6 min read · Nov 5, 2024

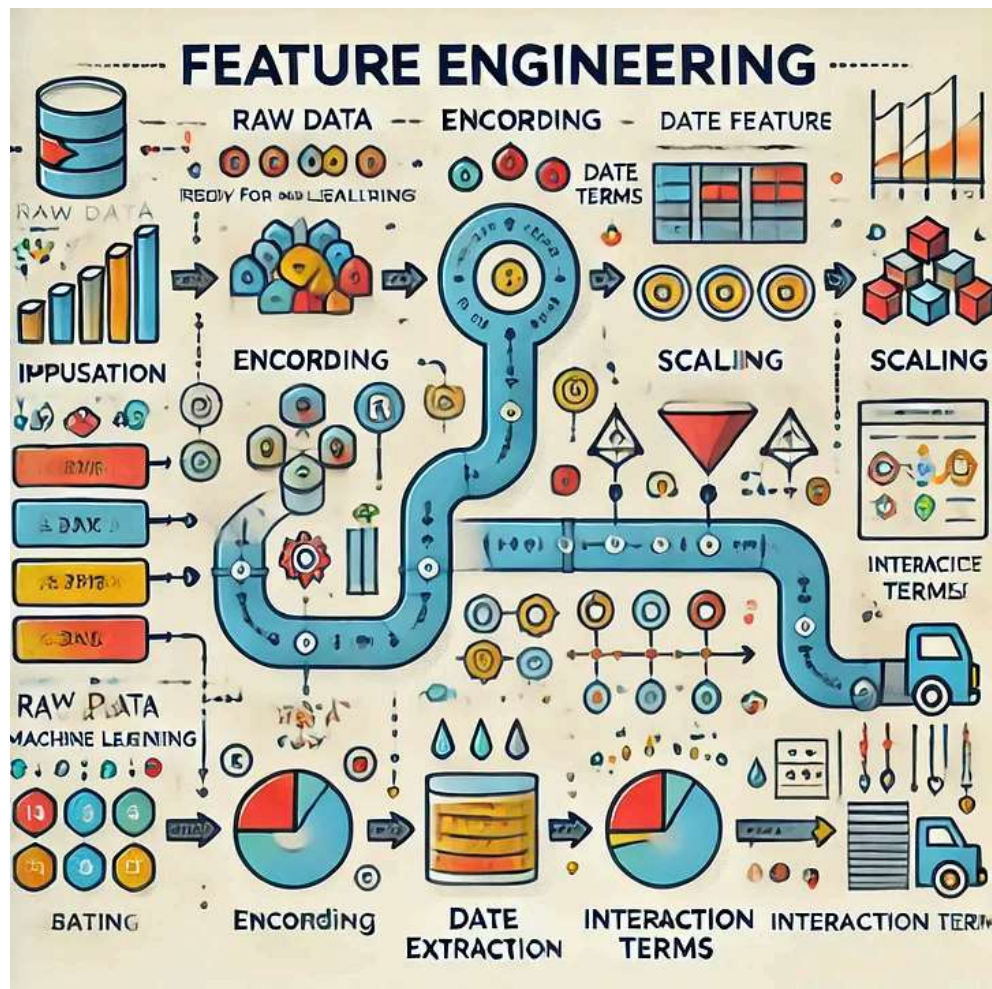


10



How to Extract Meaningful Features and Get the Most Out of Your Data

Feature engineering is one of the most critical parts of the machine learning process. It refers to the process of converting raw data into meaningful features that could improve the accuracy of a model. In many cases, this could be the difference between having a good model and a great one; it is related to how well your features are quality compared to the results you obtain in machine learning. This guide will cover the basics of feature engineering so that beginners understand the importance and how to get started using practical techniques.



1. What Is Feature Engineering?

Feature engineering is the process whereby domain knowledge is applied to retrieve features or attributes from the raw data. The outcome of these extracted features now serves as inputs into models of machine learning, ultimately improving the model's predictions. In essence, therefore, feature engineering transforms this messy, unstructured nature of data into structured forms that can be understood and, more importantly, learned about by an algorithm of the machine learning model.

2. Why Is Feature Engineering Important?

Good feature sets enable the machine learning models to extract complex patterns and interaction in relationships within the data.

Features well engineered: Ensure an accuracy increase in the model with the model being offered all relevant cleaned up and structured data.

Help to reduce model complexity, which sometimes allows you to use simpler models that are easier to interpret and faster to run.

Make the model more interpretable, since engineered features usually reflect real-world phenomena or patterns.

3. Feature Engineering Workflow

Feature engineering is usually an iterative process. Here is a basic workflow to guide you:

Understand the Problem: Clearly determine what kind of business or research problem you're dealing with and what type of prediction is required. You know that the problem determines and guides you to what sort of features are in effect useful for your use.

Explore the Data: Deeply explore your dataset with knowledge of data types, ranges, distributions, and percentage missing values.

Identify Relevant Features: Peruse through your columns or attributes in the dataset, with ideas on what may or may not be relevant when affecting the target variable.

Transform and Engineer Features: Apply a mix of techniques to create new features or transform existing ones.

Test the model with the new features and see whether performance has improved or not.

4. Key Feature Engineering Techniques

Some of the core techniques you could apply to transform raw data into more meaningful features are as follows:

a. Missing Values

Missing values usually greatly affect the performance of a model. Here is how you can deal with them:

Mean/Median Imputation: Replace missing numeric values with the mean or median of the column.

Mode Imputation: Use mode-most frequently occurring value-for categorical data

Indicator Variables: Introduce a new binary feature which will indicate if a value was missing or not.

b. Encoding Categorical Variables

Machine learning models want numerical input so the categorical variable needs to be transformed into numerical formats:

One-Hot Encoding: Each category is mapped into a column of zeroes and ones, useful when the categorical variables are of low cardinality.

Label Encoding: map every unique category to an integer number. This can be effective with ordinal categories: low, medium, and high.

Target Encoding: replacing categories by the mean value of the target variable per category, which is somewhat more complex and should be applied very carefully to prevent data leakage.

c. Scaling and Normalization

For features that occur on different scales (such as age and income), scaling will standardize them so they have the same scale:

Standardization: Subtract the mean and divide by the standard deviation (useful for normally distributed data).

Normalization: Rescale data to fall within a given range, such as 0 to 1. Useful when data has outliers.

d. Feature Transformation (Log, Square Root, etc.)

If your data is not normally distributed, feature transformation can help your model fit the data better:

Log Transformation: It compresses the range and hence reduces the dominance of very large values. It is helpful for a positive skew data.

Square Root Transformation: This reduces the skewness but is less extreme than log transformation.

e. Polynomial Features

It is sometimes enough to augment linear models with polynomial terms of numerical features to be able to capture complex interactions:

Quadratic Terms: Use squared terms for each feature to identify non-linear effects.

Interaction Terms: Take the product of two features to capture interaction between two.

f. Binning or Discretization

Categoricals-Continuous variable to be turned into discrete categories or bins. Example: For example, age is one feature that may be sensible categorical features representing age groups: 0–20, 21–40, etc. Binning makes the model simple but sometimes good for interpretation.

g. Date and Time Features

You can get more useful features if you have a date or time variable in your dataset.

Year, Month, Day, Hour: Extract date-time components for seasonality patterns.

Day of Week/Weekend Indicator: Converts days into weekdays or weekends, which can be useful in retail data.

Time Since: Measures time since a particular event, such as the first purchase date.

5. Feature Selection: Choosing the Right Features

You have created features, now you need to select those that are most impactful. Common feature selection techniques are:

Correlation Analysis: Identify highly correlated features and remove redundancies.

Univariate Feature Selection: Rank features by statistical tests and take the top.

Recursive Feature Elimination (RFE): Train the model many times and eliminate the lowest-ranked feature each time.

6. Practical Example: Feature Engineering on a Sample Dataset

Let's work through a simple example using a test data set that predicts house prices. Here is a portion of data attributes you might have:

Price (target variable)

Size (sq ft): Continuous numerical variable

Bed Rooms: Categorical integer

Sale Date: Date of sale

Location: categorical variable (neighborhood)

Step 1: Missing Values

Use the median size to fill in any missing size available.

Step 2: Encode categorical variables

Apply one-hot encoding for Location by having a dummy feature for every unique neighborhood

Step 3: Date Transformation

Extract the Year of Sale, Month of Sale and Day of Sale from Date of Sale

Step 4: Feature Scaling

Standardize Size to have mean 0 and standard deviation 1.

Step 5. Polynomial and Interaction Features

Produce an interaction feature which is the element-wise multiplication of Size and Number of Bedrooms so that we are able to capture that house size influences price partly through the quantity of bedrooms.

Step 6: Model and Evaluate

Verify that the engineered features have utility by fitting a model using the new feature set and then ascertaining the resulting accuracy.

7. Conclusion: Feature Engineering as a Continuous Process

Feature engineering is quite the art rather than an exact science. It applies to creativity, iteration in understanding of both data as well as a problem at hand and creativity and iteration. Well-crafted features can drive simple models to perform pretty well while the worst features kill the best model. Try various feature engineering techniques, see their impact, and remember that feature engineering is an ongoing process, often requiring domain knowledge and experimentation.

Mastering the basics of feature engineering will bring you closer to developing high-performance models and makes you a proficient data scientist!

In Plain English

Thank you for being a part of the [In Plain English](#) community! Before you go:

- Be sure to **clap** and **follow** the writer 🙌
- Follow us: [X](#) | [LinkedIn](#) | [YouTube](#) | [Discord](#) | [Newsletter](#) | [Podcast](#)
- [Create a free AI-powered blog on Differ.](#)
- More content at [PlainEnglish.io](#)

Data Science

Machine Learning

Deep Learning

Python

NLP



10



Published in Artificial Intelligence in Plain English

Follow

14K Followers · Last published 12 hours ago

New AI, ML and Data Science articles every day. Follow to join our 3.5M+ monthly readers.



Written by Ritesh Gupta

Follow

3.6K Followers · 28 Following

Data Scientist, I write Article on Machine Learning| Deep Learning| NLP | Open CV | AI Lover ❤️

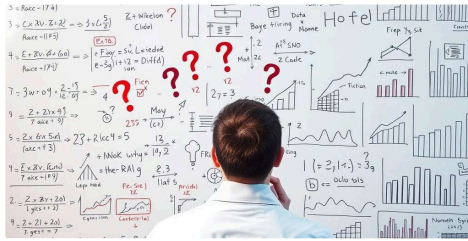
No responses yet



What are your thoughts?

Respond

More from Ritesh Gupta and Artificial Intelligence in Plain English




 Ritesh Gupta

Can You Handle These 25 Toughest Data Science Interview Questions?

The role of a Data Scientist demands a unique blend of skills, including statistics, machine...

★ Sep 25 🖱 211 💬 7




 In Artificial Intelligence in Plain En... by Andrew B...

New KILLER ChatGPT Prompt—The “Playoff Method”

Super powerful prompt for ChatGPT—01 Preview

★ Sep 27 🖱 5K 💬 96




 In Artificial Intelligence in Plain ... by Antony Matt...

Only 1% Chat GPT users know these Secret Prompts

These can 10X the Quality of your Chat GPT Responses

Oct 16 🖱 2.3K 💬 27



 In Python in Plain English by Ritesh Gupta

7 GitHub Repos to Transform You into a Pro ML/AI Engineer

Hands-On Guides, Tools, and Frameworks to Fast-Track Your AI Journey

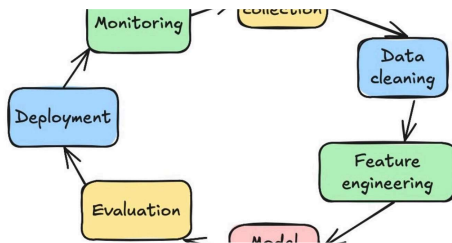
★ Nov 5 🖱 143 💬 1



See all from Ritesh Gupta

See all from Artificial Intelligence in Plain English

Recommended from Medium

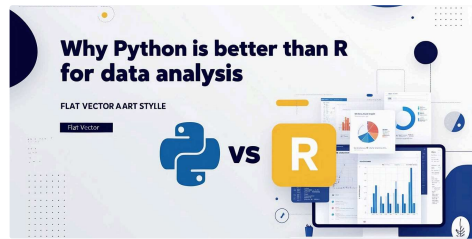


tds In Towards Data Science by Haden Pelletier

Every Step of the Machine Learning Life Cycle Simply Explained

A comprehensive guide to the ML life cycle with examples in Python

★ 3d ago 🖱️ 193 💬 3 📖



PY In Python in Plain English by Mayur Koshti

Why Python is Better than R for Data Analysis

Python and R for data analysis would involve breaking down each relevant aspect, feature...

★ Nov 15 🖱️ 34 💬 7 📖

Lists



Predictive Modeling w/ Python

20 stories · 1691 saves



Practical Guides to Machine Learning

10 stories · 2057 saves



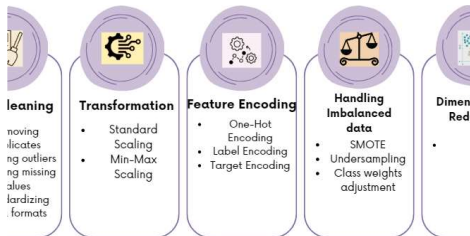
Natural Language Processing

1839 stories · 1461 saves



Coding & Development

11 stories · 920 saves



AI In Code Like A Girl by Niveetha Manickavasagam

Top 5 Data Preprocessing Techniques: Beginner to...

A Comprehensive Guide to Clean, Transform, and Optimize Data Effectively

★ Nov 18 🖱️ 61 💬 1 📖

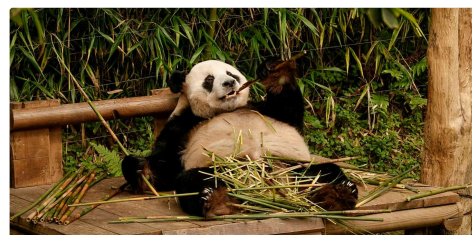



AI In Artificial Intelligence in Plain English by Ritesh Gu...

From Jupyter to Production: Deploying Machine Learning...

Turn your Jupyter Notebook experiments into production-ready applications with this...

★ Oct 24 🖱️ 123 📖



 Nilimesh Halder, PhD

How to Prepare Data for Machine Learning Models in Python?

Data preparation is a crucial step in building effective machine learning models, especiall...

★ Oct 29 🖱️ 2 💬 1 

 In Level Up Coding by Leo Anello

23+ Pandas Data Preprocessing Examples

A Comprehensive Guide to Data Cleaning, Transformation, and Preparation Using...

Jul 5 🖱️ 68 

See more recommendations