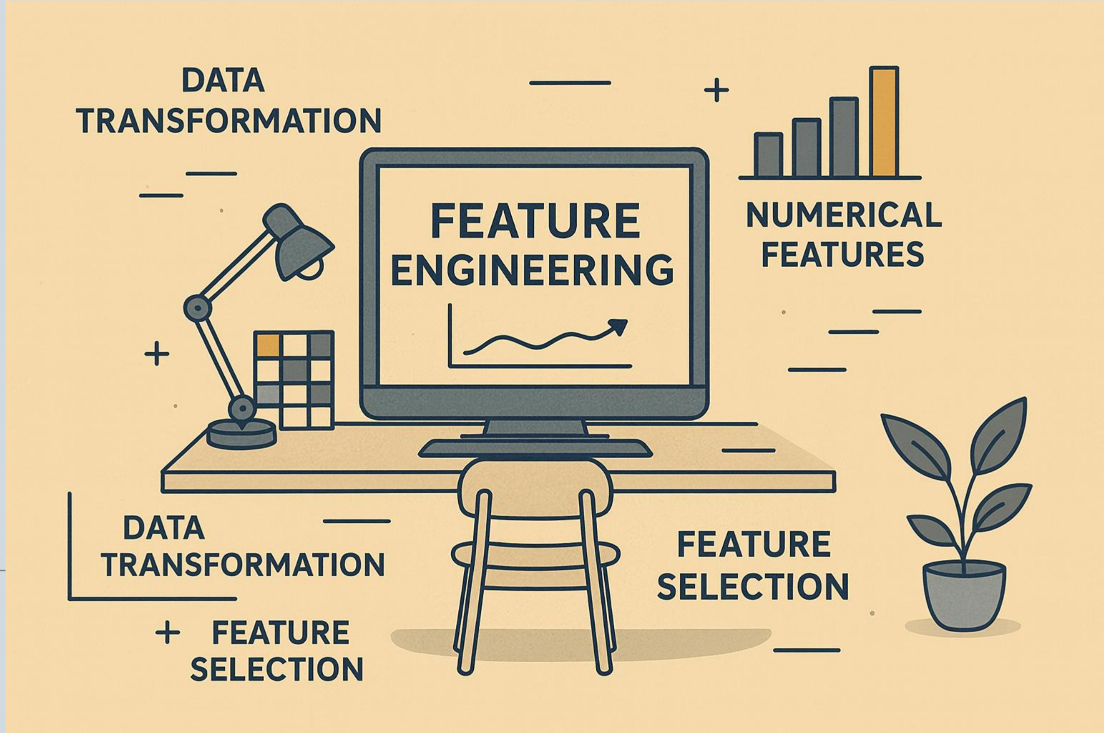
# Feature Engineering

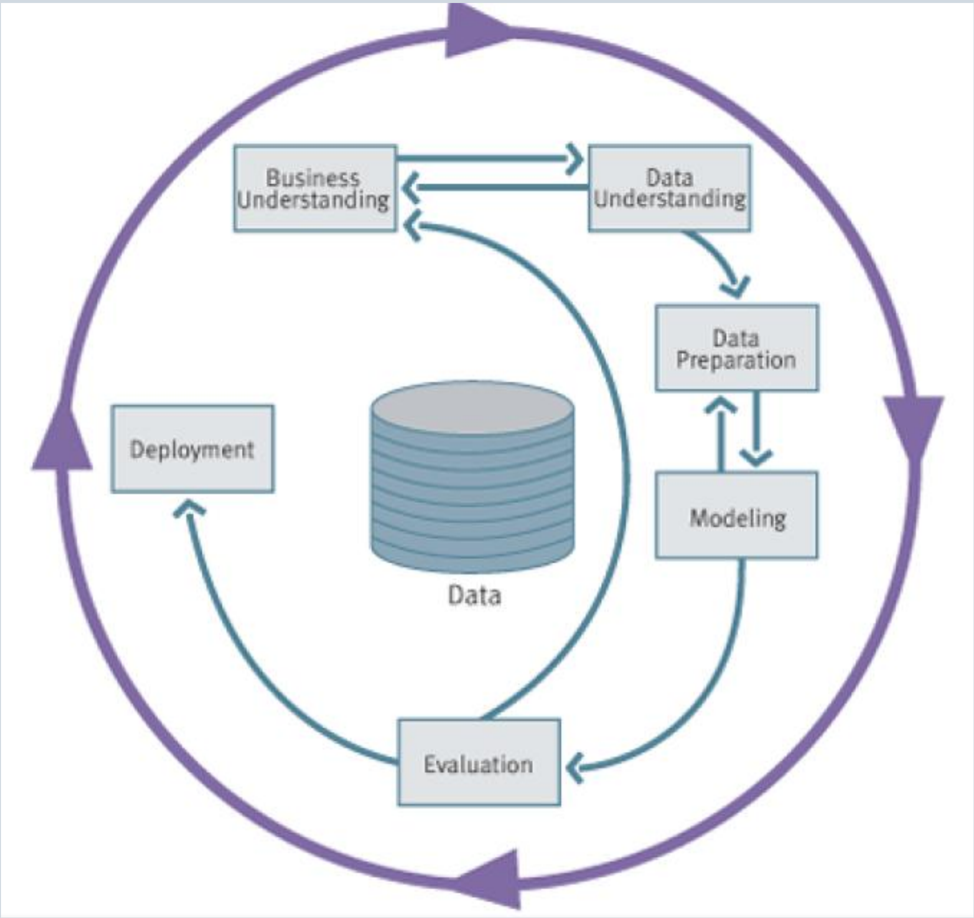



DR. ZVI BEN AMI

# Agenda

- Why feature engineering matters?
- Feature engineering:
  - Data imputation
  - Categorical encoding
  - Transformations & scaling
  - Handling outliers
  - Feature creation & selection
- Demo (notebook snippets)

# Data Mining Process

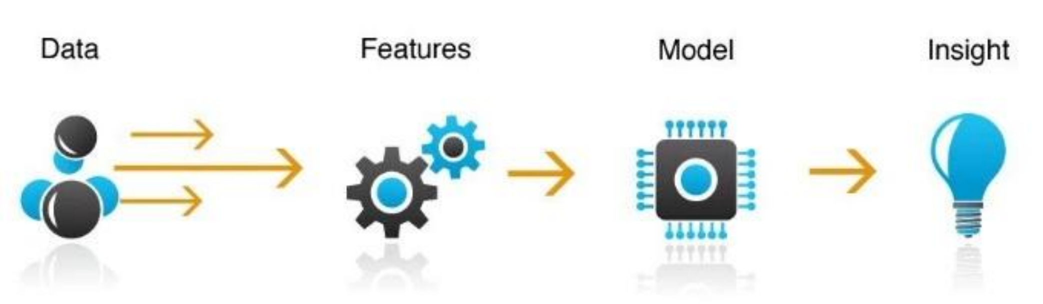



**CRISP-DM in details:**
https://www.ibm.com/docs/it/SS3RA7_18.3.0/pdf/ModelerCRISPDM.pdf

Cross Industry Standard Process for Data Mining (Shearer, 2000)

# Feature Engineering

- Feature Engineering is the process of transforming raw data into meaningful inputs that improve a machine learning model's performance.
- It combines domain knowledge, data preprocessing, and creativity.
- Motivation:
  - Raw data rarely fits directly into ML models.
  - Good features often matter more than the algorithm.
  - High-quality features often have more impact than the choice of algorithm.
  - Helps models:
    - Learn patterns more easily
    - Reduce noise & bias
    - Generalize better




Source: https://www.analyticsvidhya.com/blog/2018/08/guide-automated-feature-engineering-featuretools-python/

# Feature Engineering

**Feature_engineering.ipynb**

- Types of feature engineering:
  - Imputation
  - Categorical encoding
  - Transformations
  - Handling Outliers
  - Features manipulations
    - Discretization
    - Splitting
    - Transformations
  - Creating new features

Some references:

- https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/
- https://www.projectpro.io/article/8-feature-engineering-techniques-for-machine-learning/423
- https://www.analyticsvidhya.com/blog/2021/09/complete-guide-to-feature-engineering-zero-to-hero/

# MISSING VALUES

What can we do if we are missing some values our data?

- Remove the whole observation (row/vector)?
- Remove the whole feature (column)?
- What else can we do?

| carat | cut | color | clarity | depth | table | price | x | y | \ |
|---|---|---|---|---|---|---|---|---|---|
| 0.23 | Ideal | E | SI2 | 61.5 | 55.0 | NaN | 3.95 | 3.98 | |
| 0.21 | NaN | E | SI1 | 59.8 | 61.0 | 326.0 | 3.89 | 3.84 | |
| 0.23 | Good | E | NaN | 56.9 | 65.0 | 327.0 | 4.05 | 4.07 | |
| 0.29 | Premium | I | VS2 | 62.4 | 58.0 | 334.0 | 4.20 | 4.23 | |
| 0.31 | Good | J | SI2 | 63.3 | 58.0 | 335.0 | 4.34 | 4.35 | |

# DATA IMPUTATION

We can try to guesstimate the missing values.

How?

If we have missing numerical values, we can use:

- Mean
  - df.fillna(df.mean())
- Median
  - df.fillna(df.median())
- Mode
  - df.fillna(df.mode())


We can also add a feature that indicates whether any imputation occurred in the correct feature vector and let the model learn this when fitting the data.

# CATEGORICAL ENCODING

Machine learning models require a numeric input and generate a numeric output.

How can we handle categorical (string) variable?

Binary features :

- 1/0

Ordinal features:

- encodes the values as integer.

Nominal features:

- One-Hot Encoding: encodes the values as a binary vector array.
  - pd.get_dummies(**df.cut**)
  - pd.get_dummies(df.cut, **prefix=cut'**)
- Dummy Variable Encoding: same as One-Hot Encoding, but one less column.
  - There is some redundancy in One-Hot encoding
  - pd.get_dummies(df.cut, **drop_first=True**)
- pd.**concat**(**[df, embarked_dummies]**, axis=1)

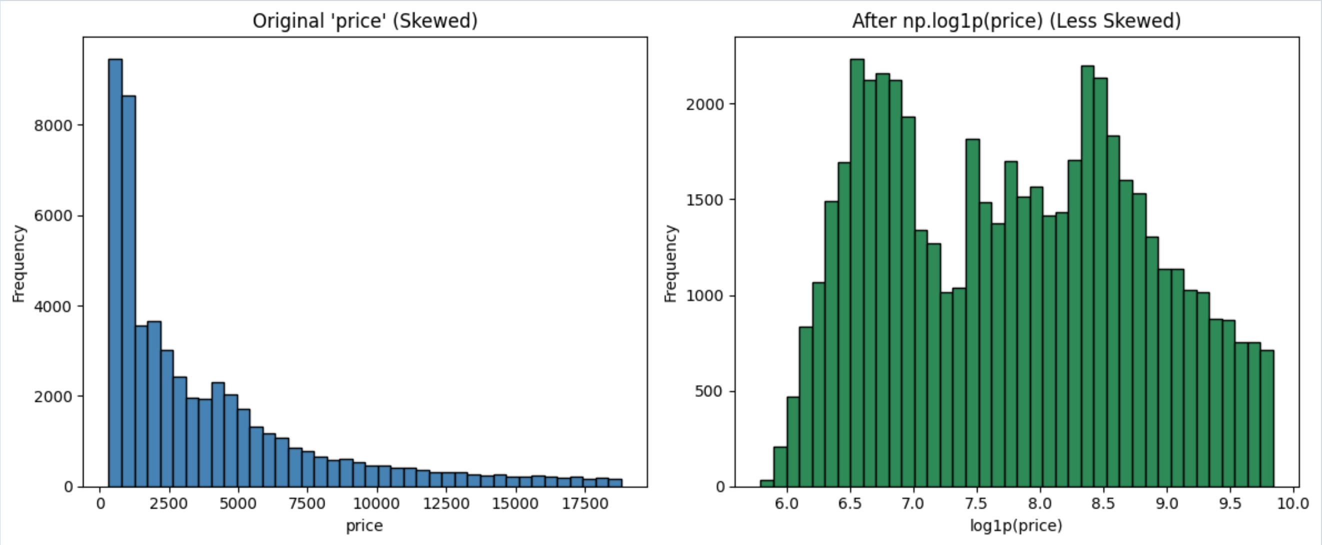
# MANIPULATION AND TRANSFORMATIONS

## Manipulation:

- Discretization (grouping into bins)
- Splitting
- Marge

## Transformations:

- Logarithme – log(x)
- Square root transformation – sqrt(x)
- Inverse transformation – 1 / x
- Exponentiel transformation – exp(x)



# HANDLING OUTLIERS

Identify outliers

- Common method: use X standard deviations

Handling outliers

- Removal
- Treating as missing values
- Capping

```
factor = 3
upper_lim = data_remove_outliers['price'].mean () +
data_remove_outliers['price'].std () * factor
lower_lim = data_remove_outliers['price'].mean () -
data_remove_outliers['price'].std () * factor
```

# CREATING NEW FEATURES

We can use two or more existing features and create a new variable.

E.g.,

- If have a feature of the total budget spent by a client in our company and the number of items she purchased, then we can create a new feature of the average budget spent by the client on an item (or vice-versa).
- If we have a list of product and a list of locations in which they were produced, we can create a features of locations ratio.
- We can generate a binary feature indicating whether a provider is reliable or not, by setting a set of rules for that feature, such as if the provider had over 10 cancelations, 5  late deliveries, and 15 missing items in stock then he is not reliable.

# Preprocessing Pipeline

- Pipelines in Machine Learning
  - A sequential container for preprocessing + modeling steps.
  - Ensures transformations are fit only on training data, then applied consistently to validation/test.
  - Keeps code clean, reproducible, and leakage-free.

```
pipe = Pipeline([
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler", StandardScaler()),
    ("model", LogisticRegression())
])
```

- Why Pipelines Matter:
  - Prevent data leakage (imputer/scaler never sees test data).
  - Bundle all preprocessing & model steps into one object.
  - Simplify cross-validation and grid search.
  - Make deployment easier — the pipeline becomes the full workflow.

# Key Takeaways

- Turns raw data → useful inputs for ML.
- No “one-size-fits-all” feature engineering.
- Iterative, domain-driven.
- Techniques:
  - Imputation (fill missing values)
  - Categorical encoding (one-hot, label)
  - Transformations (np.log1p(price) to reduce skew)
  - Handling outliers (remove, cap, treat as missing)
  - Feature creation (e.g., price/carat)
- Why it matters?
  - Good features often improve performance more than the choice of model.

# THANK YOU FOR LISTENING

ZVI.BENAMI@MAIL.HUJI.AC.IL