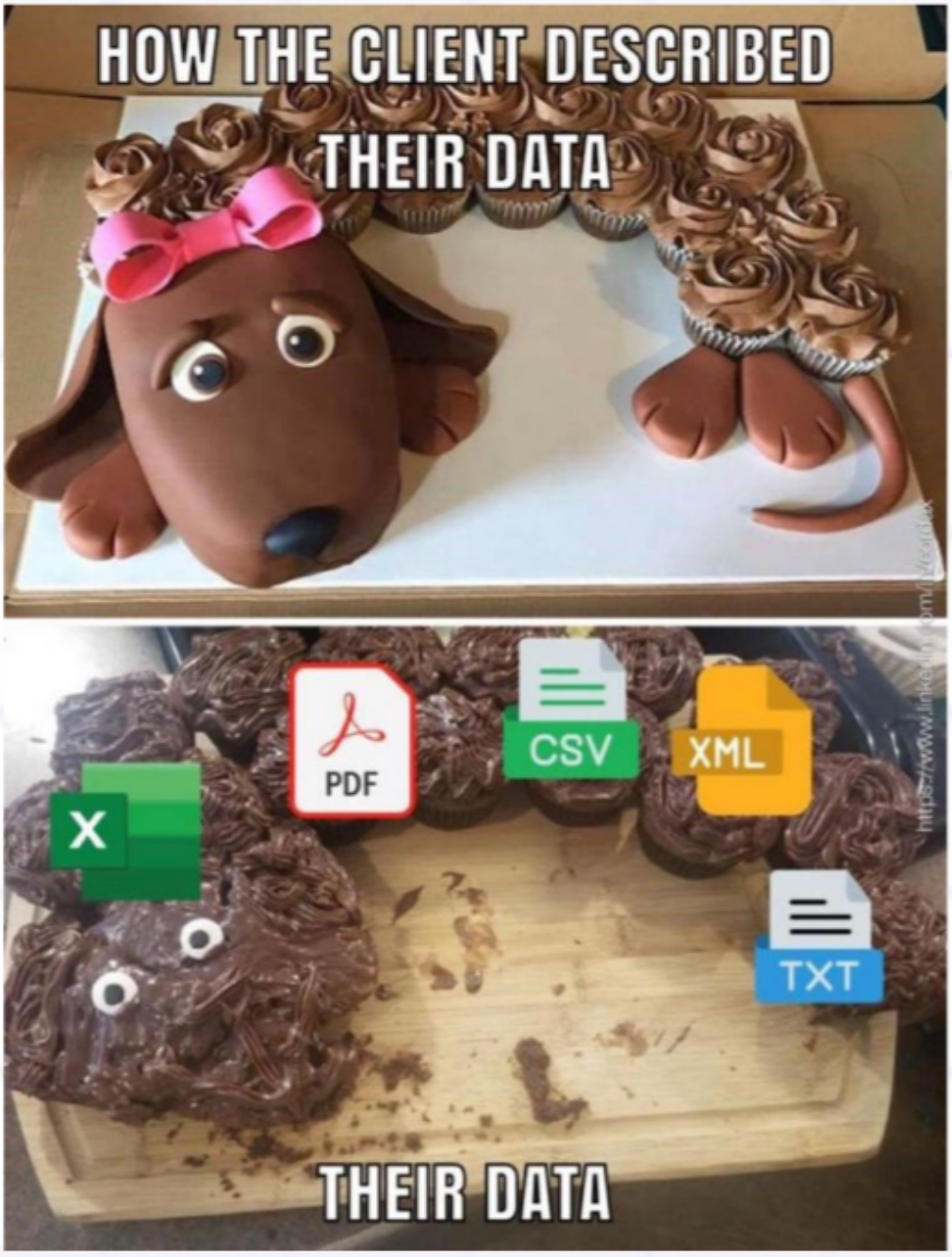
Machine Learning Model Set Up

Data Preparation



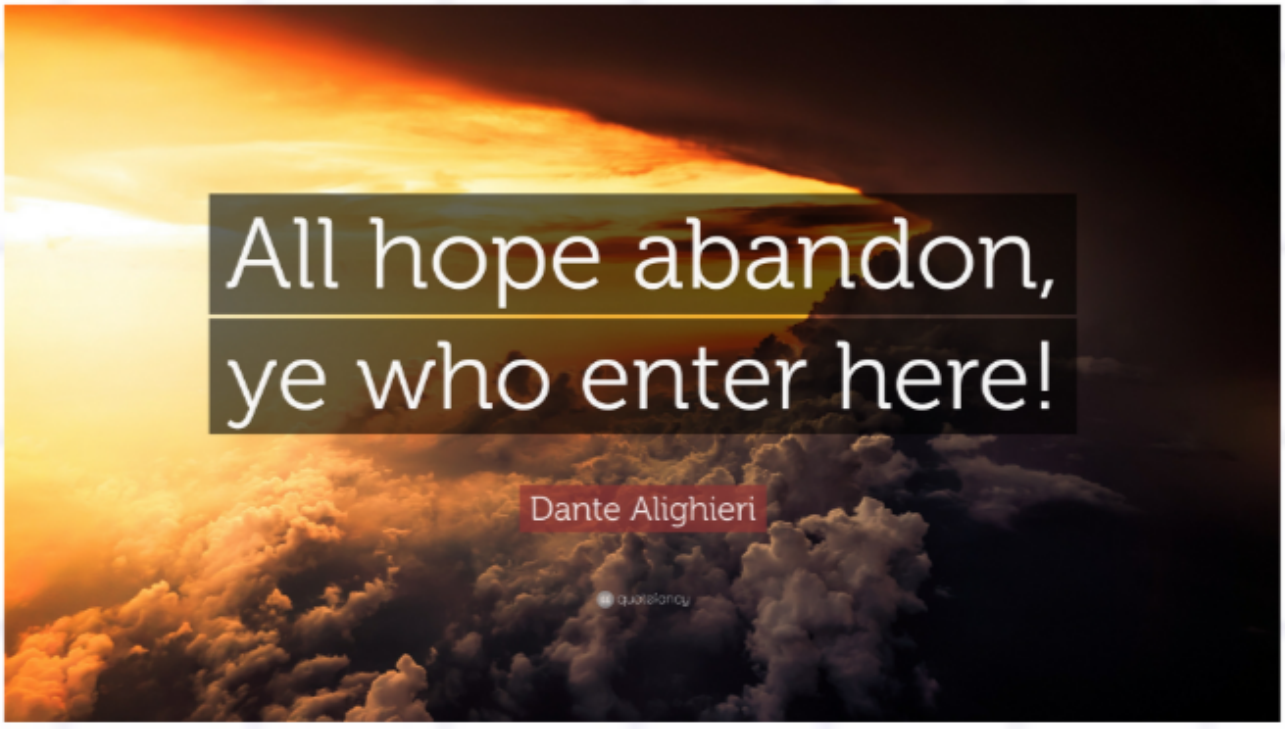

Julia Lenc

No wonder this is an iconic picture for data preprosessing



Julia Lenc

But at least there are 8 stages and not 12 circles



Welcome to Data Scientist's Inferno!

8 Stages of Data Preparation

Cleansing

↓

Labeling

↓

Handling imbalanced datasets (if applicable)

↓

Transformation (if applicable)

↓

Encoding

↓

Scaling and Normalization

↓

Feature selection

↓

Dimensionality reduction

Julia Lenc

Stage 1: Data Cleansing

1.1 Handling missing values

- Removing raws and columns
- Dropping duplicates
- Imputation:
  - Replacement: mean, median, mode, previous, next, moving average; total sample data or by segment
  - Prediction: k-NNs, interpolation (time series)
  - Assigning "unknown" (categorical variables)

Filter to identify blanks. **Pivot Tables** to review the gaps
Data Validation to set rules for complete datasets during data entry
Power Query for imputation options: Fill Down, Replace Values, etc.

WHERE IS NULL to filter and handle missing rows or columns
Adding columns with **defalut values** for conditional statements
COALESCE or CASE statements for imputation, e.g. averages

Power BI

Power Query tools: **Replace Values, Fill Down**, etc.
Build **calculated columns** to handle missing values: IF(ISBLANK(...))

pandas: .isnull().sum(), .fillna(), .dropna()
numpy: np.isnan(), np.nan_to_num(), np.nanmean()
sklearn.impute: IterativeImputer, KNNImputer, SimpleImputer

Julia Lenc

Stage 1: Data Cleansing

1.2 Addressing outliers

- Detection methods:
  - Z-score: define thresholds, e.g. > 3 σ
  - Interquartile Range (IQR), e.g >Q3+1.5×IQR or <Q1−1.5×IQR
  - Thresholds coming from domain expertise
- Handling outliers:
  - Removal
  - Caping or transforming, e.g. Winsorizing, logarithmic
  - Keeping. Applying robust methods, e.g. sklearn's RobustScaler

MEAN, MEDIAN, QUARTILE to compute thresholds such as IQR
Conditional Formatting or **Filters** to flag potential outliers visually
Group By to calculate ranges per group and identify unusual data

Write queries to identify outliers based on domain-specific thresholds
CASE WHEN or filtering for bounding outliers using Z-scores or IQR
AVG, STDDEV to establish acceptable bounds

Calculated columns with threshold logic to flag outliers
Chart visuals (**scatter or box plots**) to visualize outliers

pandas or **SciPy** to calculate IQRs, Z-scores, or generate histograms
Visualization libraries (**matplotlib or seaborn**) to identify anomalies

Julia Lenc

Stage 1: Data Cleansing

1.3 Fixing data errors and inconsistencies

- Common errors:
  - Typos: “Coce” vs. “Coke”
  - Wrong formats: "2022-02-01" vs "01-02-2022", date as string
  - Inconsistent units: liters vs milliliters
  - Duplicates
- Fixing errors:
  - String replacement: whitespaces, name standards, Regex
  - Format standardization: float, integer, currency, time, text, etc.
  - Unit standardization: diving liters by 1000, etc.
  - Removing duplicates

Text Functions (like **LEFT, RIGHT, TRIM**) for cleaning text fields
Identify and remove duplicates with the **Remove Duplicates** tool
Find & Replace for standardizing inconsistent labeling

String operations: **LOWER(), UPPER(), TRIM()** to standardize text fields
Deduplicate data using **GROUP BY** or **DISTINCT**
Regex functions for validating or transforming patterns directly

Power Query to transform text fields (**Trim, Clean**)
DAX measures or queries to remove duplicates **within visualization**

pandas for text transformations and deduplication:
.str.lower(), .drop_duplicates() or Regex within the re library.
fuzzywuzzy to clean small typos or find near matches

Julia Lenc

Check if the data is clean

Numeric data

1. **Statistics: mean, median, std dev, min, max, quartiles**
   ! A negative value ifonly positives are expected (age, income)
   ! Extremely high standard deviation
   ! Mean != Median for symmetric datasets

2. **Data visualization (anomalies):** histograms, boxplot, scatterplot

3. **Domain expertise:** sales from last 12 months, share evolution

Text data

1. **Check Consistency and Standardization**: naming, time formats

2. **Frequency analysis:**
   ! Too many unique values in a "country" field
   ! Unexpected terms in product categories

Julia Lenc

Stage 2: Data Labeling

What is data labeling?

Assigning meaningful labels or categories to raw data (images, text, audio, structured data) to make it usable for machine learning models.

Key approaches

1. Manual Labeling

- Domain experts or annotators
- Tools: Labelbox, Amazon SageMaker Ground Truth

2. Semi-Automated Labeling

- Prebuilt models for initial annotation, then manual refinement
- Tools: Python libraries (spaCy for text, OpenCV for images)

Basic annotation setup: **dropdown** menus and **VBA** scripts

Rule-based labeling: **CASE** statements or **joins** with pre-labeled datasets

For categorical data: **calculated columns** or labeling **rules** via **DAX** logic

pandas: rule-based labeling, using .apply()
Label-Studio: open source tool for labeling
scikit-learn: automatic labeling via clustering or similarity-based algorithms

Julia Lenc

Stage 3: Handling Imbalanced Data

What is class imbalance?

When one class significantly outnumbers another in a classification dataset, causing biased models.

Key techniques

1. **Oversampling:** add samples for the minority class
   - SMOTE: Generates synthetic samples by interpolating between existing minority samples.
   - ADASYN: Focuses on harder-to-learn minority samples
2. **Undersampling:** reduce samples from the majority class
   - Random: randomly remove majority class instances.
   - Informed: removes redundant or less informative examples
3. **Synthetic Data Generation:** create entirely **new data** points for the minority class using advanced methods like GANs (Generative Adversarial Networks). The generate entirely new minority class samples for better variability.

Manual oversampling with **duplicate rows**

Query-based oversampling with **UNION ALL** or undersampling with **LIMIT CASE** or **conditions** to stratify the data

Visualize **class distributions** and filter data for manual inspection
Then **flag outliers** using calculated columns or measures

imbalanced-learn library: implement SMOTE, ADASYN or undersampling

Julia Lenc

Stage 3: Handling Imbalanced Data

What is imbalance in regression problems?

When the target variable or certain subgroups are underrepresented in the data, leading to biased predictions or poor generalization.

Key techniques

1. **Transformation for skewed targets:** reduce extreme skeweness
   - Logarithmic Transformation: handles heavy-tailed distributions.
   - Power Transformations: use Box-Cox or Yeo-Johnson to normalize.
2. **Reweighting samples:** assigns higher weights to underrepresented target ranges or groups in the loss function.
3. **Subgroup Resampling:** augments underrepresented subgroups in covariates
   - Random Oversampling: duplicates data in underrepresented subgroups
   - Domain-Driven Sampling: generates synthetic features based on domain knowledge

log or **square root** transformations on skewed target columns

Resample underrepresented subgroups using **CASE** statements
Apply aggregation/normalization functions for skewed targets

Power BI

Create **calculated measures** to normalize target distributions and visualize imbalances in subgroups

nompy: np.log() functions
scikit-learn : use sample_weight parameter in regression models
imbalanced-learn: SMOTER
scipy.stats: boxcox, yeojohnson

Julia Lenc

Stage 4: Data Transformation

What is data transformation?

Data transformation modifies the feature space or target variable to normalize distributions, handle outliers or improve interpretability.

Key techniques

1. **Log Transformation:** address long-tailed distributions
2. **Box-Cox Transformation:** for positive data requiring normalization
3. **Yeo-Johnson Transformation:** ideal for zero/negative values

Log: **LOG()** or **LOG10()**
Box-Cox: Compute **lambda** externally and plug it into the formula
=IF(lambda=0, LN(A2), ((A2^lambda) - 1) / lambda)

Determining lambda: Python (scipy.stats.boxcox()), R or trial-and-error

Log: **SELECT LOG()**
Box-Cox: Compute lambda externally and integrate via **SELECT CASE**

Log: **LOG()**

Log: **np.log()** functions from **numpy**
Box-Cox: **boxcox** function from **scipy.stats**
Yeo-Johnson: **yeojohnson** function from **scipy.stats**

Julia Lenc

Stage 5: Encoding

What is data encoding?

Encoding converts categorical variables into a numerical format to make them suitable for downstream tasks such as scaling, modeling and analysis.

Key techniques

1. **One-Hot Encoding:** for categories without ordinal relationship
   → colors, countries, brands
2. **Label Encoding / Ordinal Encoding:** for with an inherent order
   → low, medium, high performance
3. **Embeddings:** for high-cardinality categories, preserved relationships
   → Zip codes, user IDs, product IDs, text analysis

One-Hot: **dummy variables**
Label: **lookup table** with VLOOKUP or nested IF statements to assign numbers to categories

One-Hot and Label: **CASE** statements

One-Hot: **IF()** calculated columns
Label: **SWITCH()** calculated columns

One-Hot: **pandas** pd.get_dummies(df['Category'], prefix='Category')
Label: **LabelEncoder** function from **sklearn.preprocessing**
Embeddings (for deep learning): **tensorflow** or **pytorch**

Julia Lenc

Stage 6: Scaling

What is data scaling?

Scaling ensures that numerical variables are consistent across a similar range, which is important for machine learning algorithms that are sensitive to scale differences (those relying on gradient descent or distance metrics like SVMs, KNNs).

Key techniques

1. **Standardization:** for algorithms that assume normally distributed data, e.g., linear regression. Centers the data around 0 and scales using the standard deviation.
2. **Normalization (Min-Max Scaling):** models sensitive to magnitude, e.g., neural networks. Scales data to a fixed range, usually [0, 1].

Standardization: =(A1 - **AVERAGE**(A:A)) / **STDEV**(A:A)
Normalization: =(A1 - **MIN**(A:A)) / (**MAX**(A:A) - **MIN**(A:A))

SELECT statements
Standardization: (Value - AVG(Value)) / STDDEV(Value) AS
Normalization: (Value - MIN(Value)) / (MAX(Value) - MIN(Value)) AS

New columns or **Power Query**
Standardization: = ([Value] - AVERAGE('Table'[Value])) / STDEV.P('Table'[Value])
Normalization: ([Value] - MIN('Table'[Value])) / (MAX('Table'[Value]) - MIN('Table'[Value])

skikit-learn functions
Standardization: scaler = **StandardScaler()**
Normalization: scaler **= MinMaxScaler()**

Stage 7: Feature Selection

What is feature selection?

Selecting important features to remove irrelevant or redundant ones, aiming to improve computational efficiency and model performance.

Key techniques

1. **Statistical tests and domain knowledge:** ANOVA, Chi-Square, Correlation, Mutual Information.
2. **Lasso:** shrinks coefficients of less important features to zero
3. **Recursive Feature Elimination (RFE):** iteratively removes features to optimize a model's performance.
4. **Tree-Based:** Random Forest or XGBoost to rank features

Correlation Analysis (Analysis Toolpak for small datasets)

Filter features with correlation or aggregations

Lasso, RFE, Tree-Based Feature Importance

Julia Lenc

Stage 8: Dimensionality Reduction

What is dimensionality reduction?

Reducing the number of features by projecting them onto a lower-dimensional space while retaining maximal information.

Key techniques

1. **Principal Component Analysis (PCA):** converts correlated features into uncorrelated principal components, retaining the majority of variance.
2. **t-SNE:** Compresses high-dimensional data into 2D/3D for visualization while preserving local neighborhood structure.

PCA (**sklearn**)
t-SNE (**sklearn, TensorFlow**)
You can consider UMAP (**umap-learn** library), which is faster than t-SNE, but more complex.

Julia Lenc

Final Validation

Build a Sanity Check Model

Train a basic model using preprocessed dataset

Check Model Performance

For classification: accuracy, precision, recall, F1 score.

For regression: RMSE, R-Squared, MAE.

Confirm Expected Results

Are the most critical features ranked higher by your model?

Are the results consistent across the training and test sets?

Is the performance significantly better than a baseline guess?

Other methods

Peer review

Cross-validation

Test with real-world input

Automated frameworks: Great Expectations and Pandera

Tools: Pandas Profiling, Sweetviz, and D-Tale

Julia Lenc

Did you find it useful?

Save

Share

Follow

Analytics, Market Research,
Machine Learning and AI

Julia Lenc