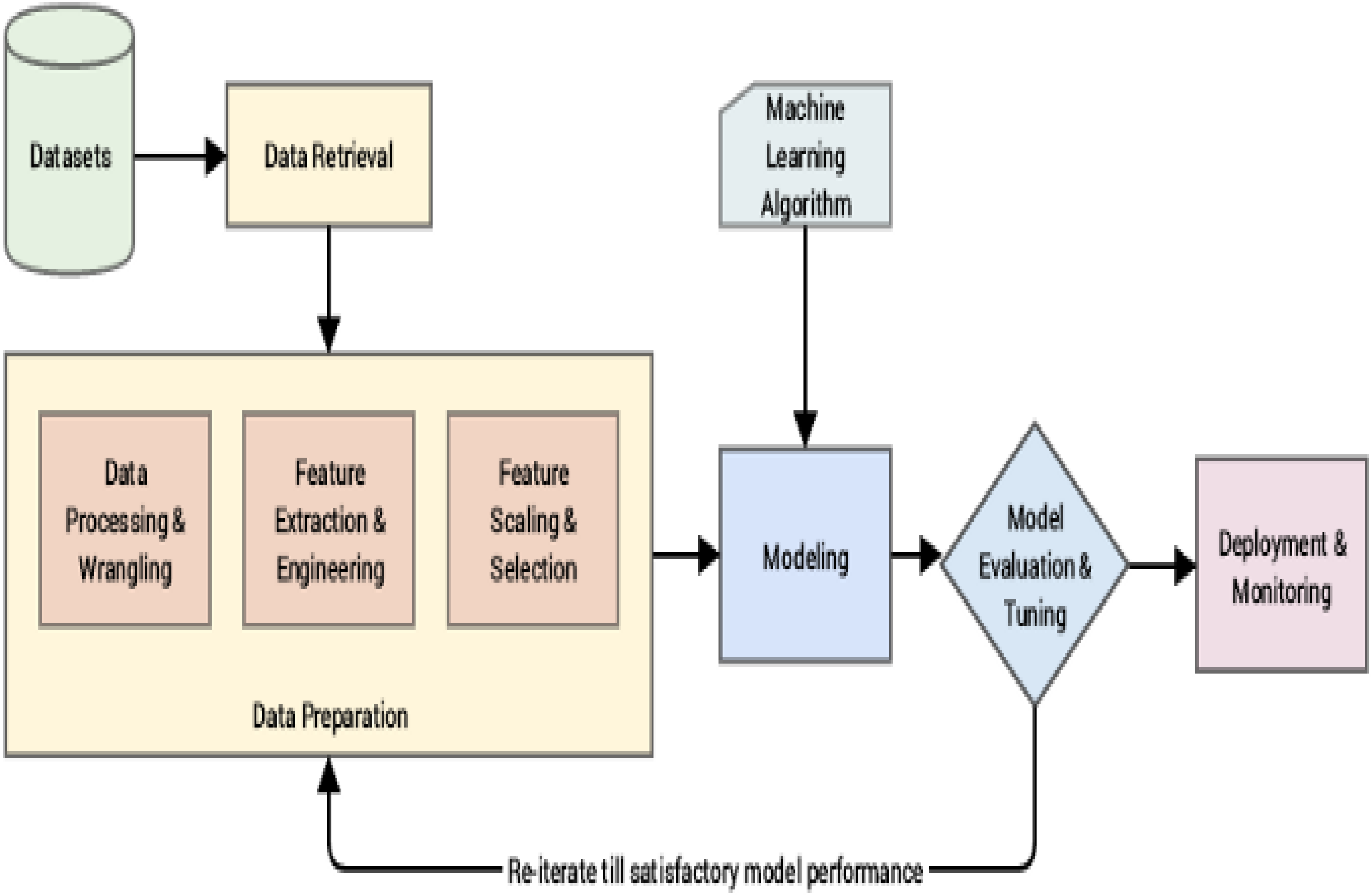
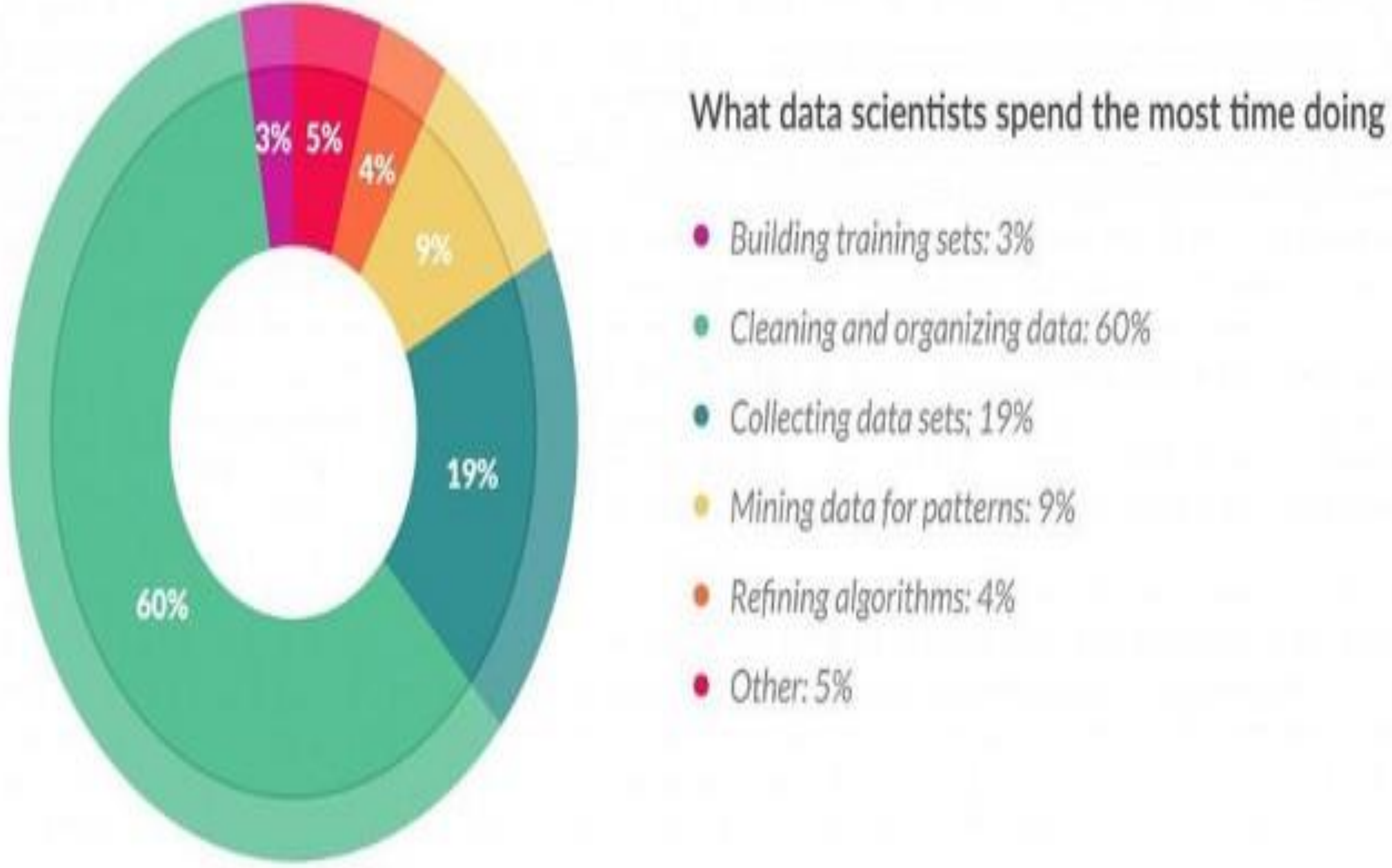
# Feature Engineering for Machine Learning


Datasets
Data Retrieval
Machine Learning Algorithm
Data Processing & Wrangling
Feature Extraction & Engineering
Feature Scaling & Selection
Data Preparation
Modeling
Model Evaluation & Tuning
Deployment & Monitoring
Re-iterate till satisfactory model performance

A standard machine learning pipeline (source: Practical Machine Learning with Python, Apress/Springer)



Source: https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/

# Motivation

*"Coming up with features is difficult, time-consuming, requires expert knowledge. 'Applied machine learning' is basically feature engineering."*

*— Prof. Andrew Ng.*

*"Feature engineering is the process of transforming **raw data** into **features** that better represent **the underlying problem** to **the predictive models**, resulting in improved **model accuracy** on **unseen data**."*

*— Dr. Jason Brownlee*

*"At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used."*

*— Prof. Pedro Domingos*

Feature engineering is another topic which doesn’t seem to merit any review papers or books, or even chapters in books, but it is absolutely vital to ML success. […]
Much of the success of machine learning is actually success in engineering features that a learner can understand.

— Scott Locklin, in “Neglected machine learning ideas”

# WHY ?

1. **Make simple model to perform much better than complex model.**

2. **Reduce model selection time.**

3. **Reduce training time by simplifying the model.**

# PANDAS

- Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

# Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

# Pandas Data Structures

- Series

Series is a one-dimensional array like structure with homogeneous data. For example, the following series is a collection of integers 10, 23, 56,

- Key Points
- Homogeneous data
- Size Immutable
- Values of Data Mutable

# Pandas Data Structures

## DataFrame

DataFrame is a two-dimensional array with heterogeneous data

Example

| Name | Age | Gender | Rating |
|---|---|---|---|
| Steve | 32 | Male | 3.45 |
| Lia | 28 | Female | 4.6 |
| Vin | 45 | Male | 3.9 |
| Katie | 38 | Female | 2.78 |

# Pandas Data Structures

Key Points

- Heterogeneous data
- Size Mutable
- Data Mutable

# Panel

Panel

- Panel is a three-dimensional data structure with heterogeneous data. It is hard to represent the panel in graphical representation. But a panel can be illustrated as a container of DataFrame.

- Key Points
- Heterogeneous data
- Size Mutable
- Data Mutable

# Data Selection

- **Data Import**

Use these commands to import data from a variety of different sources and formats.

- pd.read_csv(filename) | From a CSV file
- pd.read_table(filename) | From a delimited text file (like TSV)
- pd.read_excel(filename) | From an Excel file
- pd.read_sql(query, connection_object) | Read from a SQL table/database
- pd.read_json(json_string) | Read from a JSON formatted string, URL or file.
- pd.read_html(url) | Parses an html URL, string or file and extracts tables to a list of dataframes
- pd.read_clipboard() | Takes the contents of your clipboard and passes it to read_table()
- pd.DataFrame(dict) | From a dict, keys for columns names, values for data as lists

# Data Export

Use these commands to export a DataFrame to CSV, .xlsx, SQL, or JSON.

- df.to_csv(filename) | Write to a CSV file
- df.to_excel(filename) | Write to an Excel file
- df.to_sql(table_name, connection_object) | Write to a SQL table
- df.to_json(filename) | Write to a file in JSON format

# Viewing/Inspecting Data

Use these commands to take a look at specific sections of your pandas DataFrame or Series.

- df.head(n) | First n rows of the DataFrame
- df.tail(n) | Last n rows of the DataFrame
- df.shape | Number of rows and columns
- df.info() | Index, Datatype and Memory information
- df.describe() | Summary statistics for numerical columns
- s.value_counts(dropna=False) | View unique values and counts
- df.apply(pd.Series.value_counts) | Unique values and counts for all columns

# Selection

Use these commands to select a specific subset of your data.

- df[col] | Returns column with label col as Series
- df[[col1, col2]] | Returns columns as a new DataFrame
- s.iloc[0] | Selection by position
- s.loc['index_one'] | Selection by index
- df.iloc[0,:] | First row
- df.iloc[0,0] | First element of first column

# Data Cleaning

- Use these commands to perform a variety of data cleaning tasks.

- df.columns = ['a','b','c'] | Rename columns
- pd.isnull() | Checks for null Values, Returns Boolean Arrray
- pd.notnull() | Opposite of pd.isnull()
- df.dropna() | Drop all rows that contain null values
- df.dropna(axis=1) | Drop all columns that contain null values
- df.dropna(axis=1,thresh=n) | Drop all rows have have less than n non null values
- df.fillna(x) | Replace all null values with x
- s.fillna(s.mean()) | Replace all null values with the mean (mean can be replaced with almost any function from the statistics module)
- s.astype(float) | Convert the datatype of the series to float
- s.replace(1,'one') | Replace all values equal to 1 with 'one'
- s.replace([1,3],['one','three']) | Replace all 1 with 'one' and 3 with 'three'
- df.rename(columns=lambda x: x + 1) | Mass renaming of columns
- df.rename(columns={'old_name': 'new_ name'}) | Selective renaming
- df.set_index('column_one') | Change the index
- df.rename(index=lambda x: x + 1) | Mass renaming of index

# Filter, Sort, and Groupby

Use these commands to filter, sort, and group your data.

- **df[df[col] > 0.5] | Rows where the column col is greater than 0.5**
- **df[(df[col] > 0.5) & (df[col] < 0.7)] | Rows where 0.7 > col > 0.5**
- **df. (col1) | Sort values by col1 in ascending order**
- **df.sort_valuessort_values (col2,ascending=False) | Sort values by col2 in descending order**
- **df.sort_values([col1,col2],ascending=[True,False]) | Sort values by col1 in ascending order then col2 in descending order**
- **df.groupby(col) | Returns a groupby object for values from one column**
- **df.groupby([col1,col2]) | Returns groupby object for values from multiple columns**
- **df.groupby(col1)[col2] | Returns the mean of the values in col2, grouped by the values in col1 (mean can be replaced with almost any function from the statistics module)**
- **df.pivot_table(index=col1,values=[col2,col3],aggfunc=mean) | Create a pivot table that groups by col1 and calculates the mean of col2 and col3**
- **df.groupby(col1).agg(np.mean) | Find the average across all columns for every unique col1 group**
- **df.apply(np.mean) | Apply the function np.mean() across each column**
- **nf.apply(np.max,axis=1) | Apply the function np.max() across each row**

# Statistics

Use these commands to perform various statistical tests. (These can all be applied to a series as well.)

- df.describe() | Summary statistics for numerical columns
- df.mean() | Returns the mean of all columns
- df.corr() | Returns the correlation between columns in a DataFrame
- df.count() | Returns the number of non-null values in each DataFrame column
- df.max() | Returns the highest value in each column
- df.min() | Returns the lowest value in each column
- df.median() | Returns the median of each column
- df.std() | Returns the standard deviation of each column

# Feature Engineering

- Managing missing features
- Managing Categorical Features
- Data Scaling and Normalization
- Whitening
- Dimensionality reduction
- Creating training and testing data sets

# CASE STUDY

- Print first and last five rows
- Clean data and update the CSV file
- Find the most expensive car company name
- Print All Toyota Cars details
- Sort all cars by Price column

# Categorical data

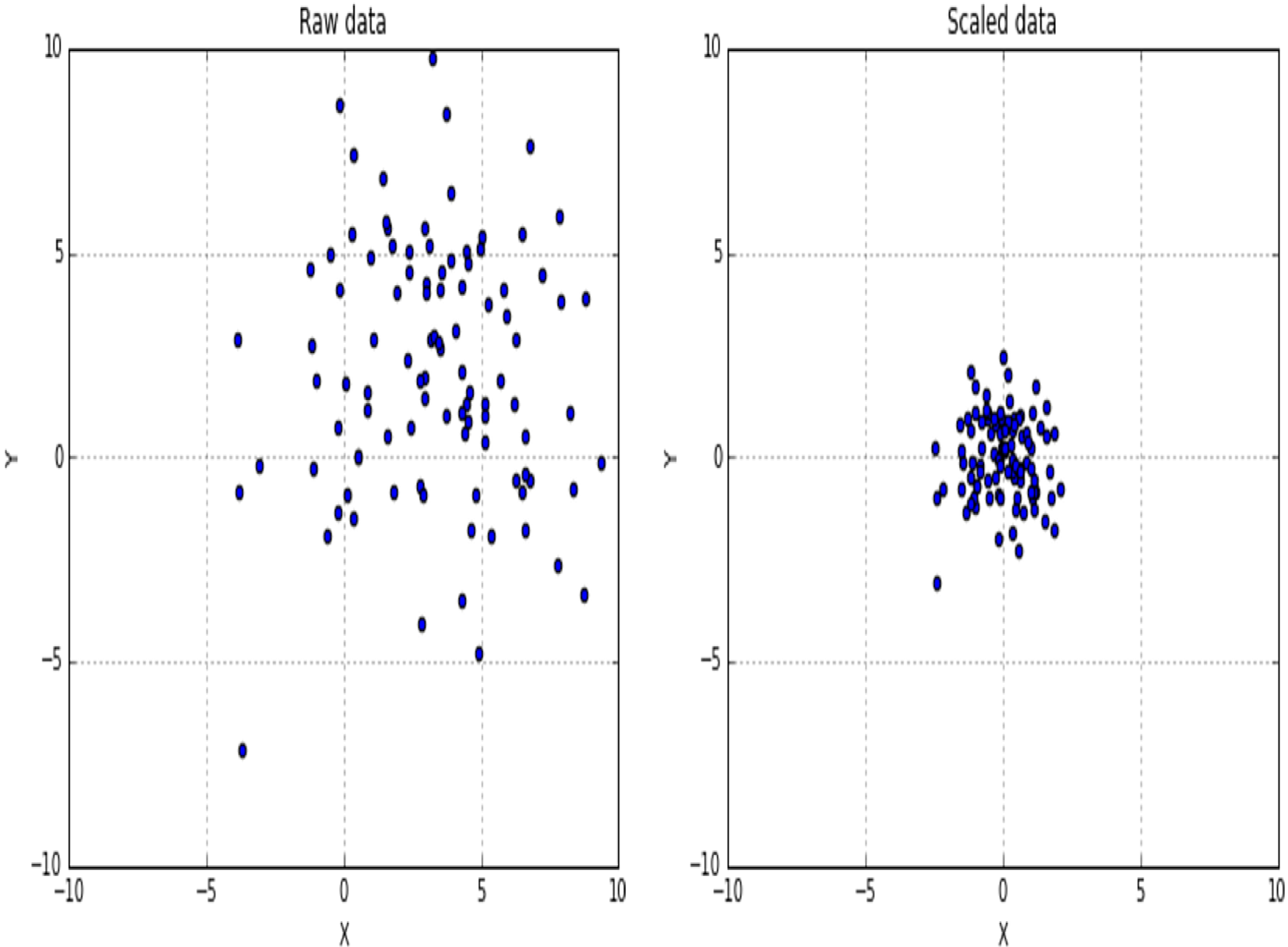
- Binary: A variable that has only 2 values. For example, True/False or Yes/No.
- Ordinal: A variable that has some order associated with it like our place example above.
- Nominal: A variable that has no numerical importance, for example color or city.

# Managing Categorical Data

- Encoding or continuation is the transformation of categorical variables to binary or numerical counterparts.

- Types
  - Binary
  - Target-based-ordinal,one-hot encoding

# Data Scaling and Normalization


Raw data
Scaled data
X
Y