ML Data Prep

Created Date: 2022-08-01

> Metadata
>
> - Title: Preparing & Cleaning Data for Machine Learning
> - Author: Andrew Jones
> - Reference: Data Science Infinity

> Links & Tags
>
> - Index: Course Note Index
> - Atomic Tag: #datascience
> - Subatomic Tags: #machinelearning #dataengineering
> - Related Notes: DSI ML Guide

Steps for Data Cleaning & Preparation

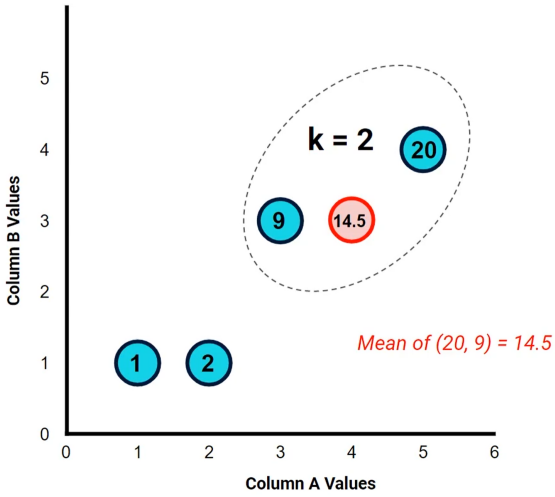
Jupyter Notebook: Data Preparation Templates

- Missing Values
- Duplicate Data & Low Variation Data
- Incorrect & Irrelevant Data
- Categorical Variables
- Outliers
- Feature Scaling
- Feature Engineering & Selection
- Validation Split (Training Set, Test or Validation Set)

Missing Values

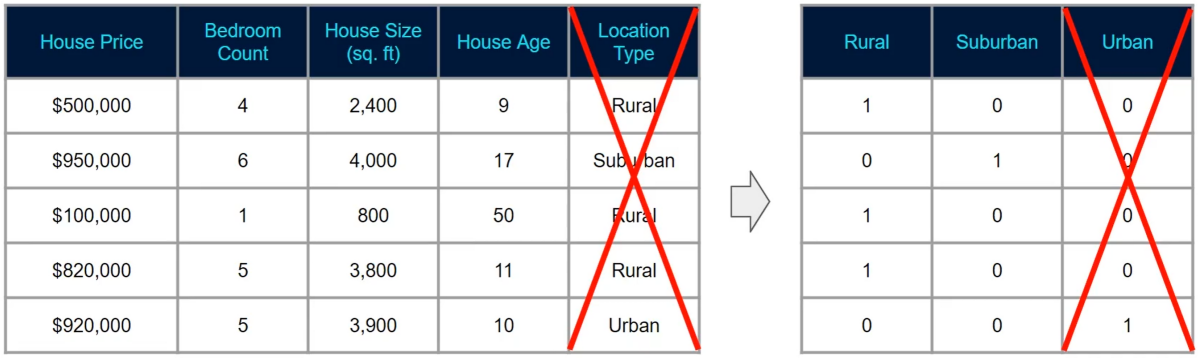
- ML models don't always know how to process NULL or missing values
- Missing values should be dropped when appropriate, or filled with predicted values
    - We can replace NULL values with the mean, median, or mode using Pandas
    - We can use a simple imputer to replace NULL values with mean, median, or mode using Scikit-Learn
    - We can use a KNN imputer to replace NULL values with nearest neighbor using Scikit-Learn
        - KNN algorithm will dynamically impute missing values using numerical data points across the entire data set (instead of a single column)
        - May give a better estimation of what the missing value will be
        - The argument n_neighbors can be used to determine how many neighbors we want the algorithm to take into consideration

A	B	C
1	1	1
2	1	2
3	3	9
4	3	14.5
5	4	20



Categorical Variables

- ML models cannot apply numerical importance to categorical variables
- Simply adding a type id can imply an order or scale that will cause inaccurate model results
- Best practice is to create dummy variables using "One Hot Encoding"

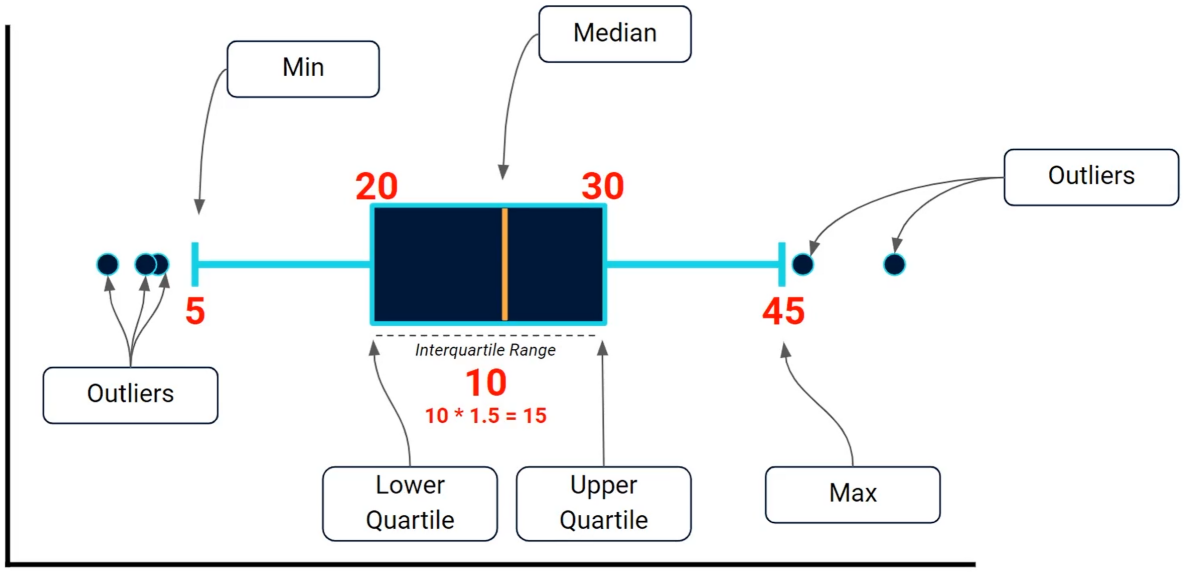
- One hot encoding is a representation of categorical variables as binary vectors
- Create new columns that represent the categories in binary
- The new columns can be used as input variables and the original column can be discarded
- Potential downfall is the "Dummy Variable Trap"
    - When input variables perfectly predict each other (multicollinearity) mainly an issue with regression modeling
    - You can drop one of the new binary category columns for each set of dummy variables created
- 

House Price	Bedroom Count	House Size (sq. ft)	House Age	Location Type
$500,000	4	2,400	9	Rural
$950,000	6	4,000	17	Suburban
$100,000	1	800	50	Rural
$820,000	5	3,800	11	Rural
$920,000	5	3,900	10	Urban

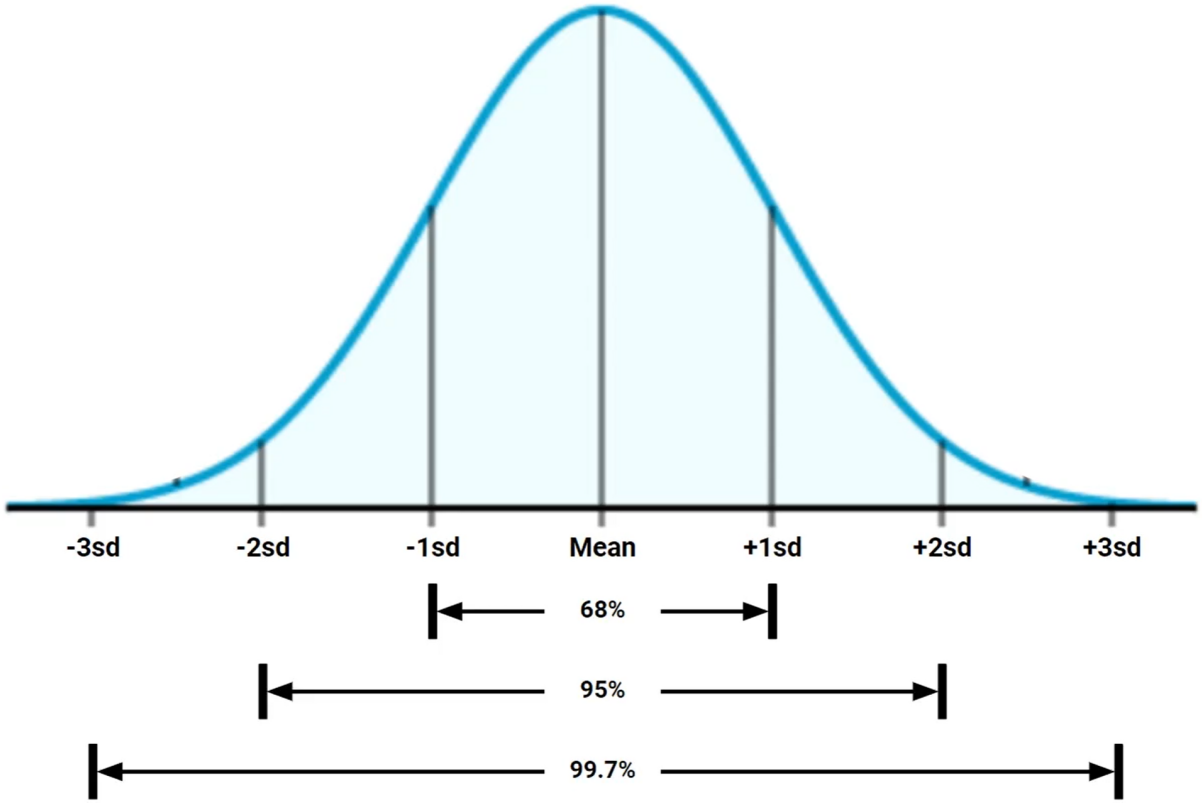
Rural	Suburban	Urban
1	0	0
0	1	0
1	0	0
1	0	0
0	0	1

Dealing with Outliers

- An outlier is any value that differs significantly from other values
- Identifying an outlier is situational
-
    - They can skew averages and ML results but in some cases they should be left in the model
- Options for handling outliers
    - Keep them
    - Remove them
    - Replace them (can be considered data manipulation)
- Identifying outliers with boxplots



- Identifying outliers with the normal distribution (outside of 3 standard deviations)



Feature Scaling for ML

- Force the values from different columns to exist on the same scale,
- Enhances the learning capabilities of the model
- Most common techniques;

    - Standardization
        - Rescales each data point to have a mean (μ) of 0 and a standard deviation (σ) of 1
        - $x_{standardized} = \frac{x-\mu}{\sigma}$
            - x Data Point
            - $\mu(X)$ Column Mean
            - $\sigma(X)$ Column Standard Deviation
    - Normalization
        - Rescales each data point so that it exists in a range between 0 and 1
        - $x_{normalized} = \frac{x-min(x)}{max(X)-min(X)}$
            - x Data Point
            - $min(X)$ Column Min
            - $max(X)$ Column Max
- Standardization is more commonly used but if your model requires all positive variables, use normalization
- This is not always a requirement and comes down to accuracy vs interpretation
    - Scaling values makes it harder to interpret the true meaning of coefficients in terms of actual values
    - Algorithms that rely on distance comparisons (K-Means, KNN) requires feature scaling

Feature Selection

- Process used to select the input variables that are most important to your ML task
- Improves model accuracy by eliminating unnecessary noise
- Lower computational cost
- Model is easier to understand and explain
- Implementing feature selection;
    - Correlation Matrix
        - Shows associations between numeric variables

    - Scores between -1 and 1, stronger relationships closer to 1
    - Input variables correlated with each other can introduce multicollinearity in regression models
    - Downside to correlation matrix is it doesn't really tell us which features to drop, we base our decisions on assumptions and pre-defined limits
- Univariate Feature Selection
    - Applies statistical tests to find relationships between the output variable and each input variable, in isolation
    - Potential downside is that it only considers variables in isolation
    - Regression

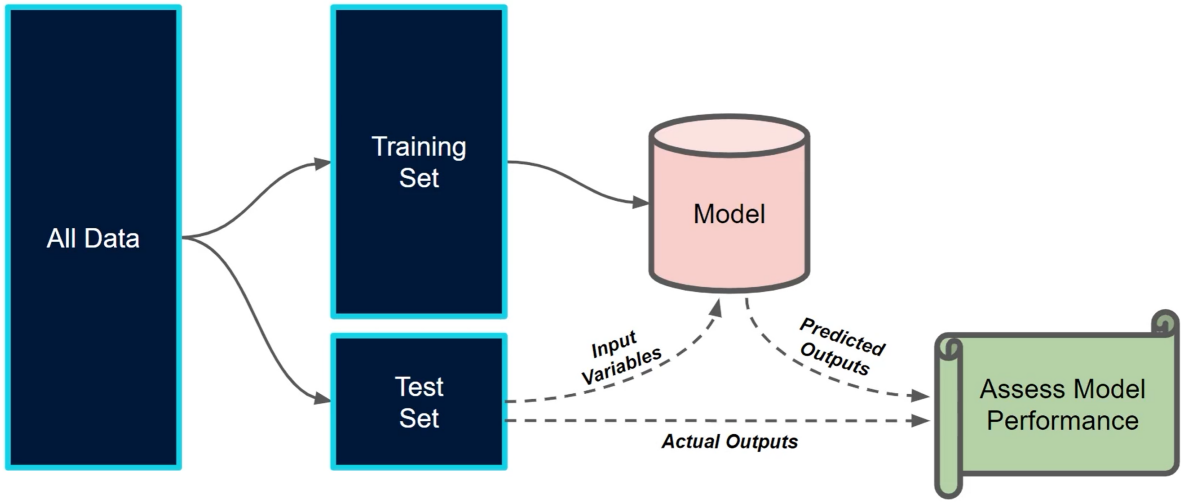
Input Variable	f-score	p-value
Input B	120	0.001
Input C	19	0.04
Input A	0.44	0.28

    - Classification

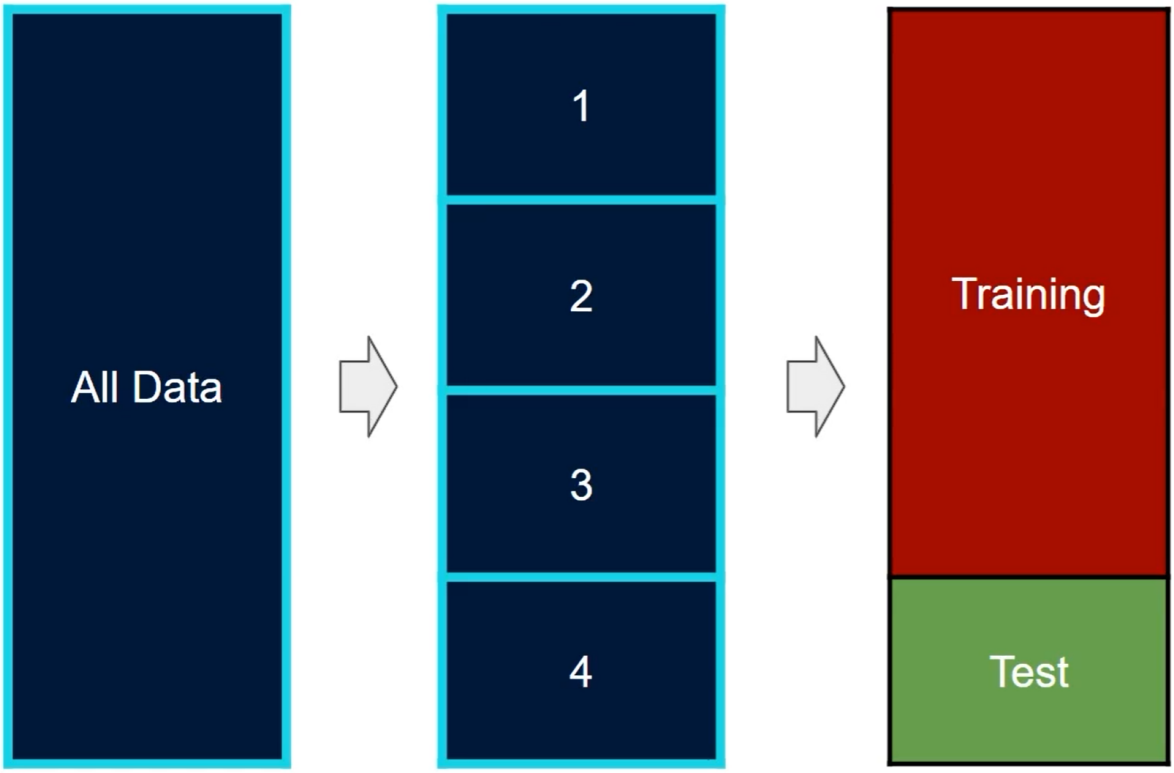
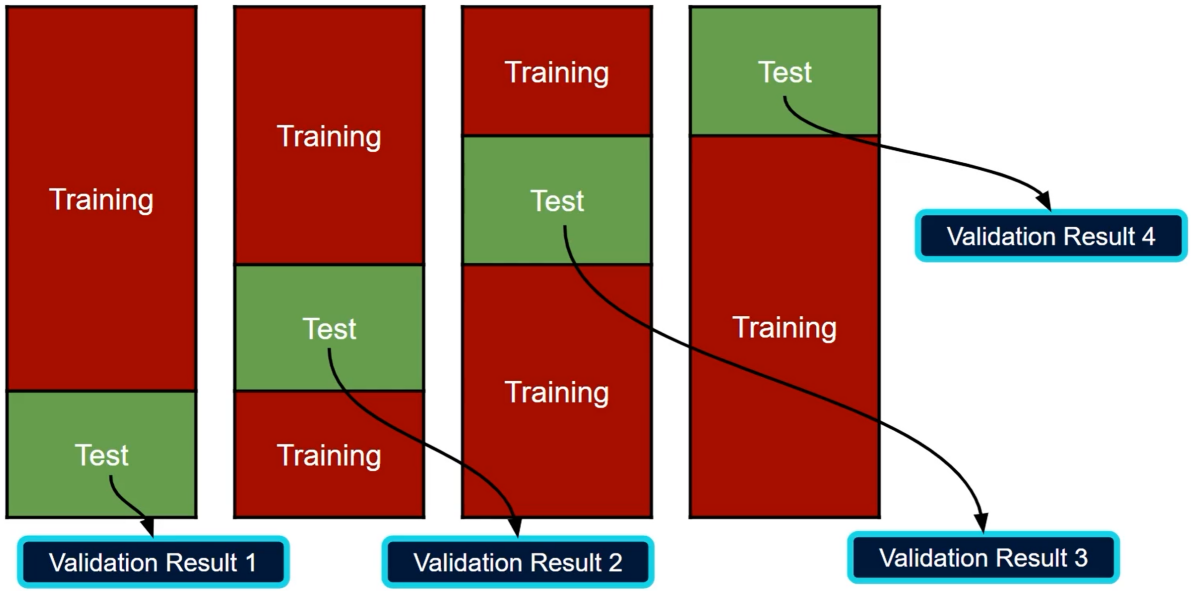
Input Variable	chi^2	p-value
Input B	7	0.008
Input C	1.7	0.09
Input A	0.21	0.38

- Recursive Feature Elimination (RFE) (usually best choice)
    - Fits a model with all input variables, then iteratively removes those with the weakest relationship until the desired number of features is reached
    - Model is re-fit after each variable is dropped
        - After a variable is dropped, the relationships will change
    - Using the desired number of features can be problematic as it is likely a "best guess"
        - Can use cross-validation to split data into different chunks and build different models
            - Algorithm will compare models to see which provided the best accuracy
            - Will tell you how many input variables should be kept and which ones they are

Model Validation & Over-Fitting

- Data should be split into two groups (order must be shuffled)
  - Training set (usually 70-80%)
  - Testing set (remaining 20-30%)
  - 

- Running models on training data can led to overfitting
  - Overfitting is where a model learns the patterns within the training data too well and results in poor performance on new data
- Another option to the above visual is Cross-Validation (K-Fold Cross Validation)
  - K denotes the number of evenly sized groups (folds) our data is divided into
  - Splitting the training and testing data so each fold is used as a testing set once
  - The results of each iteration can be compared for model accuracy

- 
All Data
1
2
3
4
Training
Test
- 
Training
Test
Validation Result 1
Training
Test
Training
Validation Result 2
Training
Test
Training
Validation Result 3
Test
Validation Result 4