



Train In Data

Variable Types

Objectives

- Understand the different types of variables
- Identify different types variables
- Examples of the different variables in a dataset

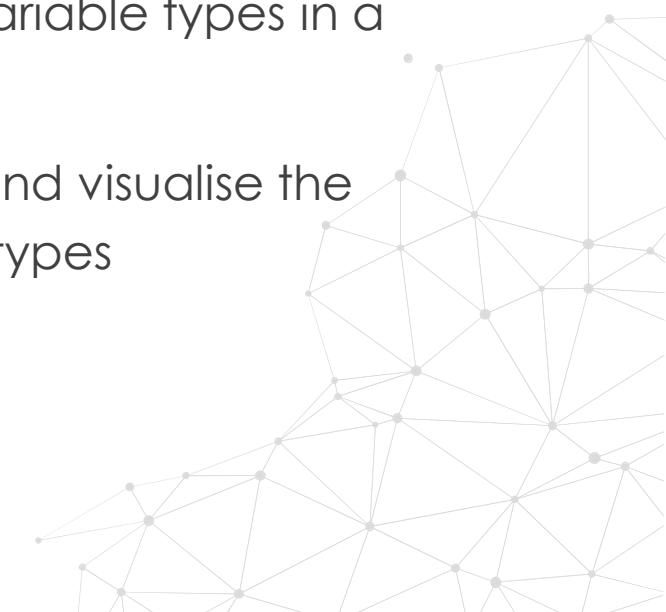


Content



For each lecture:

- Presentation and video
- Accompanying Jupyter notebook
 - Examples of the variable types in a business dataset
 - Code to identify and visualise the different variable types





What is a Variable?



• Variable

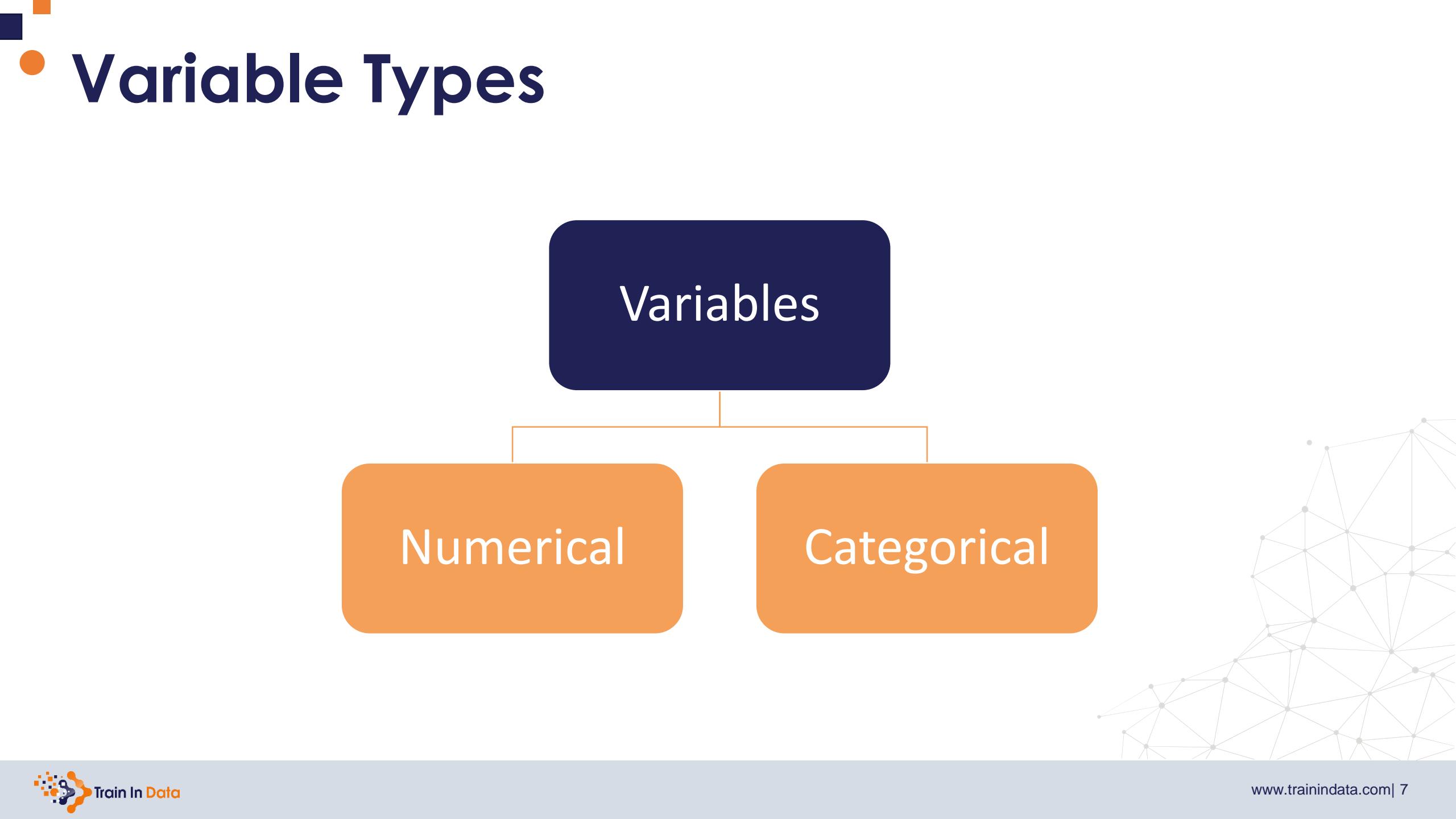
A variable is any characteristic, number, or quantity that can be measured or counted.

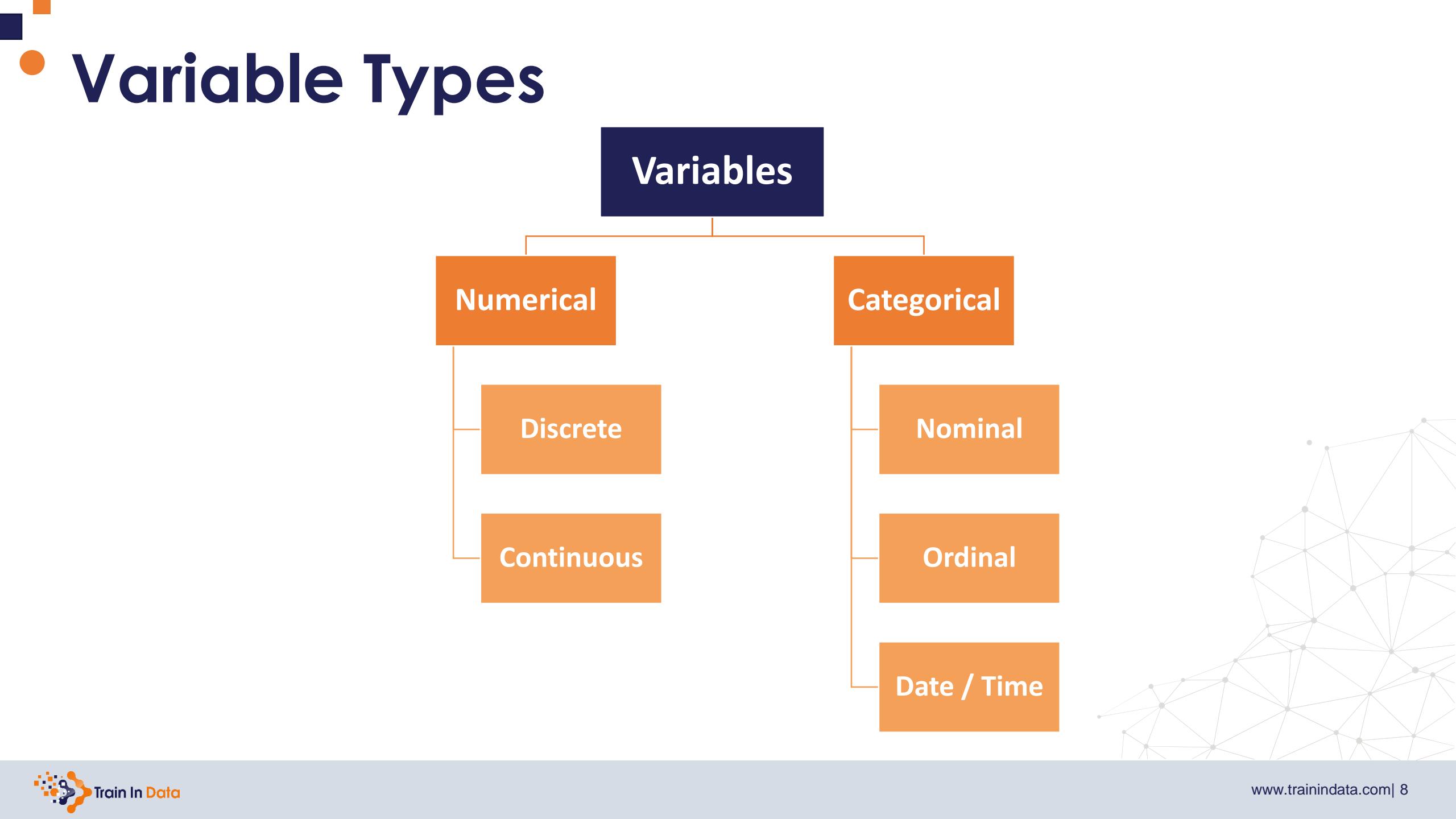


• Variable Examples

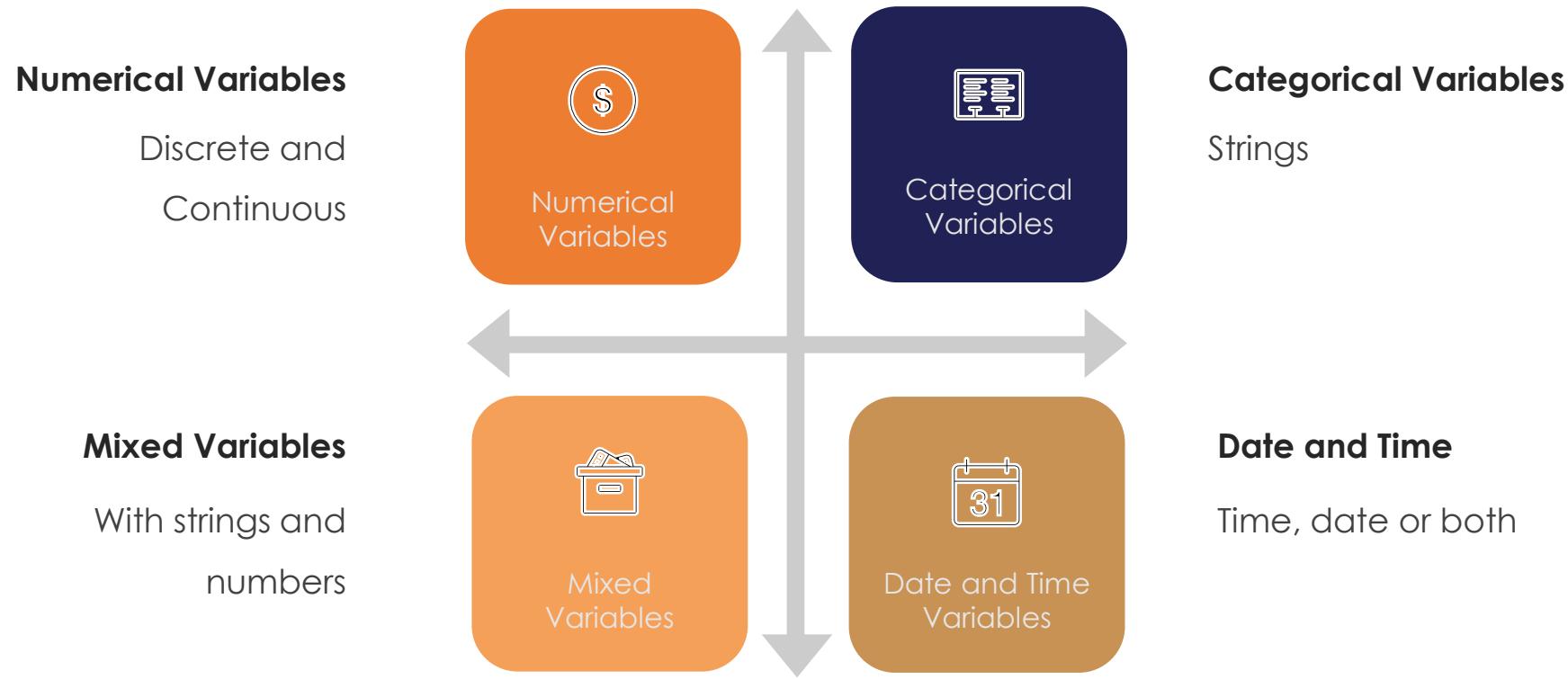
- Age (21, 35, 62, ...)
- Gender (male, female)
- Income (GBP 20000, GBP 35000, GBP 45000, ...)
- House price (GBP 350000, GBP 570000, ...)
- Country of birth (China, Russia, Costa Rica, ...)
- Eye colour (brown, green, blue, ...)
- Vehicle make (Ford, Volkswagen, ...)







In the next lectures...





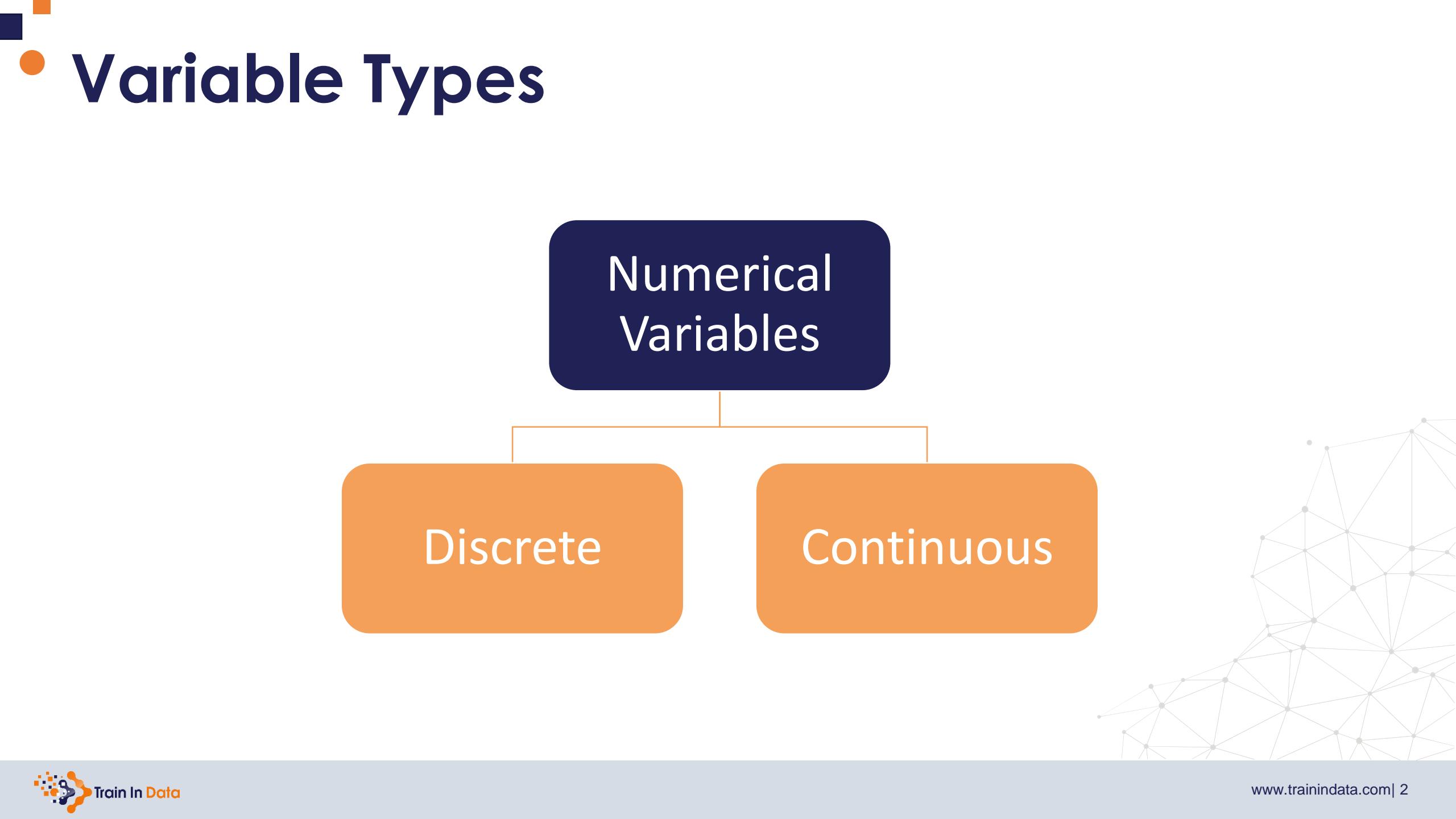
THANK YOU

www.trainindata.com





Numerical Variables



• Discrete Variables

A variable which values are whole numbers (counts) is called discrete. Examples:

- Number of items bought by a customer in a supermarket (10, 50, ...)
- Number of active bank accounts of a borrower (1, 4, 7, ...)
- Number of pets in the family
- Number of children in the family



Continuous Variables

A variable that may contain any value within some range is called continuous. Examples:

- Amount paid by a customer in a supermarket (\$32.50, \$12, \$5.20, ...)
- House price (GBP 350,000, GBP 57000, GBP 1,000,000, ...)
- Time spent surfing a website (3.4 seconds, 5.10 seconds, ...)
- Total debt as percentage of total income in the last month (0.2, 0.001, 0, 0.75, ...)

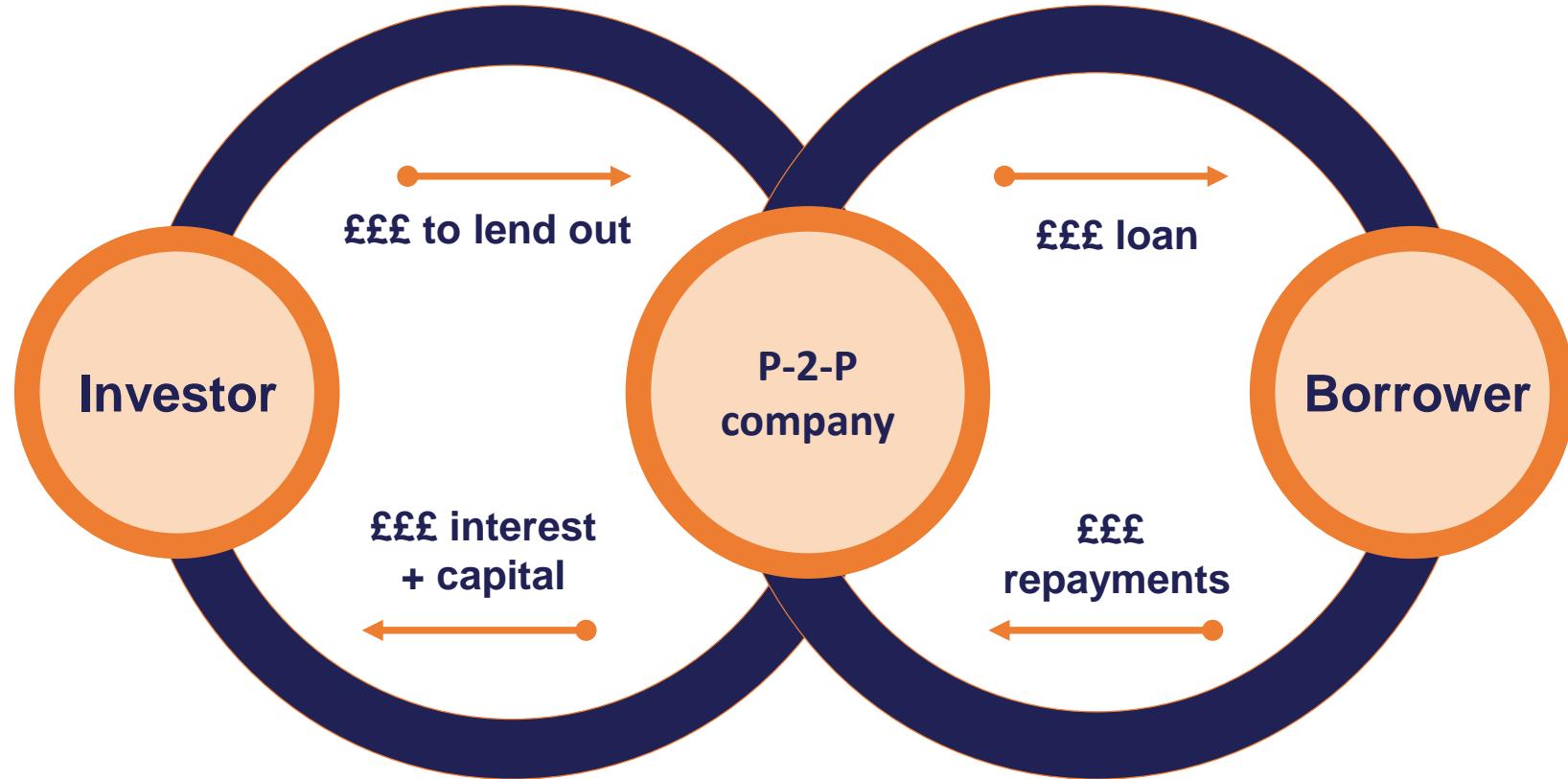


Numerical Variables

Notebook demo



Peer to Peer Finance



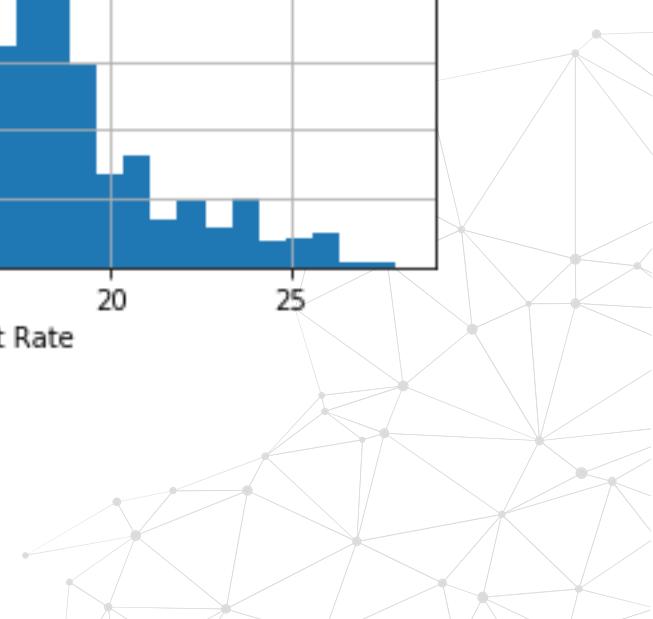
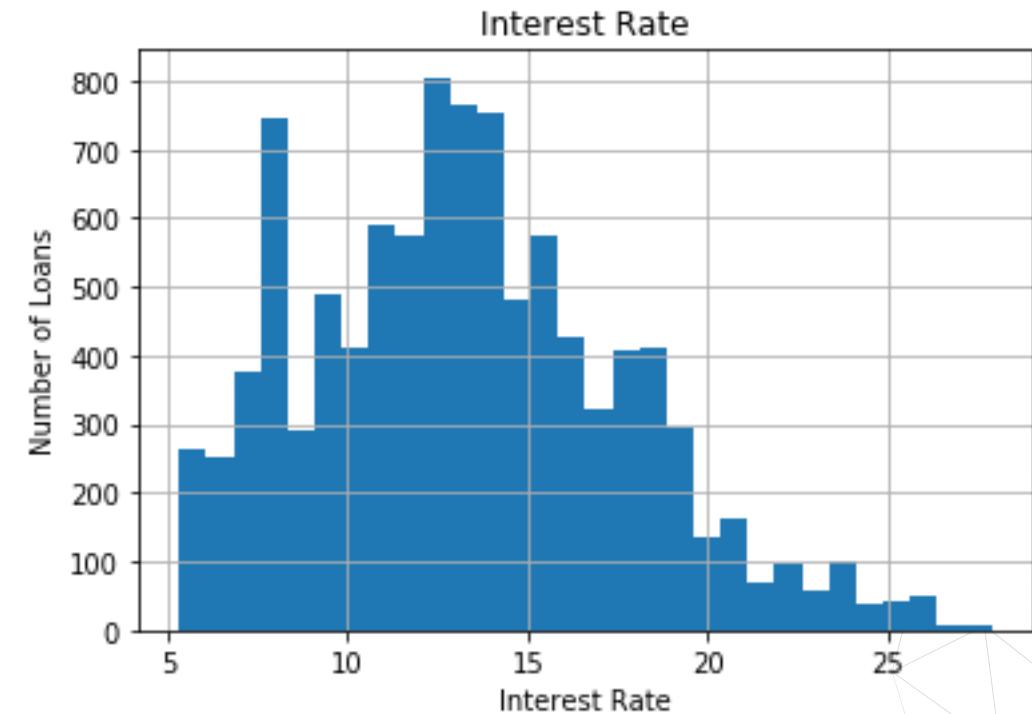
Numerical Variable Examples

- Interest rate - continuous
- Number of accounts opened in last 12 months - discrete
- Loan defaulted - binary



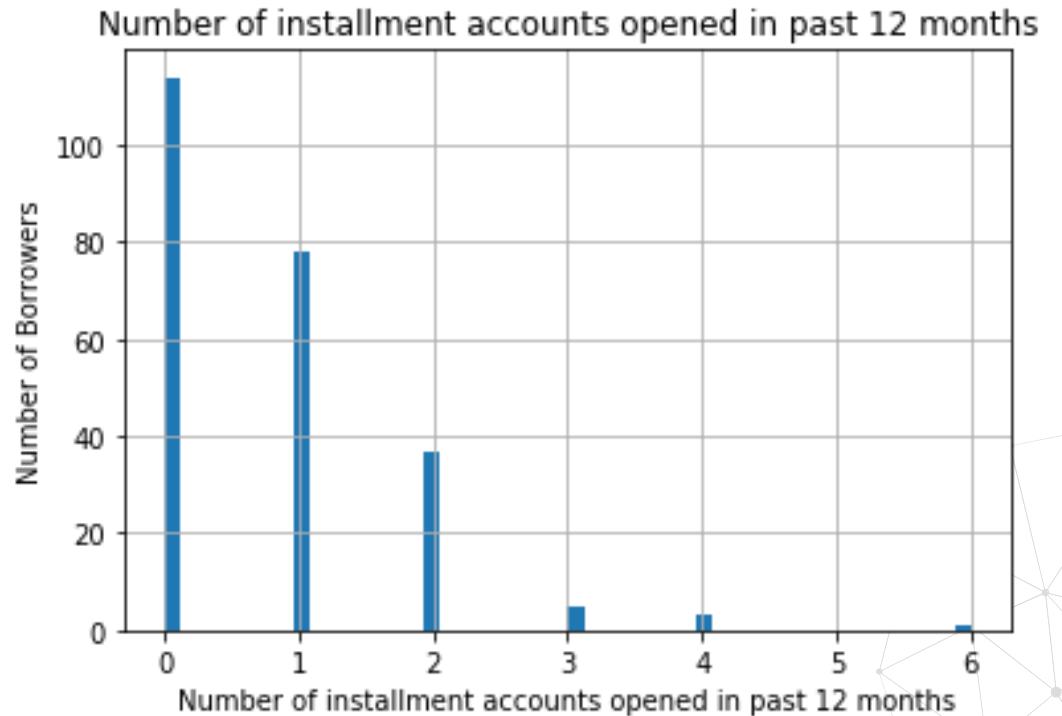
Interest Rate - Continuous

- Interest rate can take in principle any value within the range
- Example values: [15.8 , 11.67, 9.25, 6.24, 19.52.]



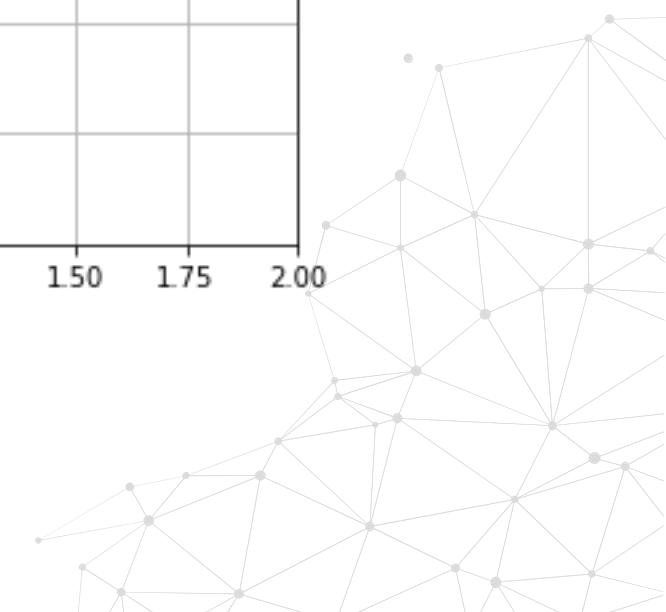
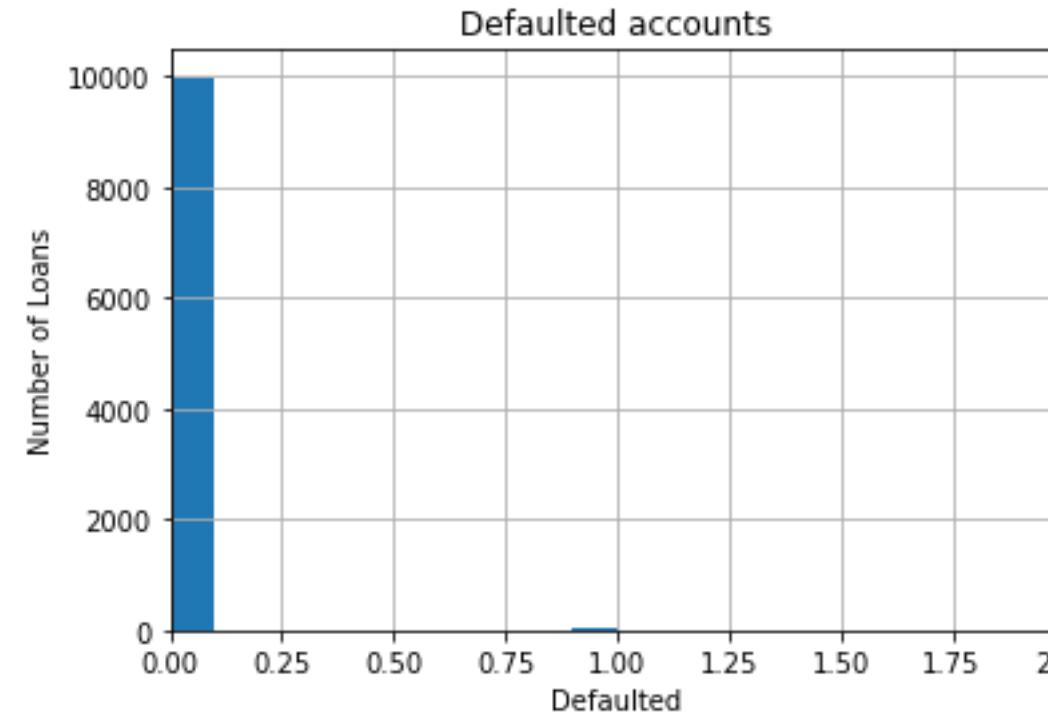
Number Opened Accounts - Discrete

- Only discrete numbers
 - Customer can have 1 account but not 1.5 accounts
- Example values: [4, 1, 6, 2, 0, 5]



• Loan Defaulted- Binary

- Takes 1 of 2 possible values
- Example values: [1, 0] or [Defaulted, Non-Defaulted]



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook





THANK YOU

www.trainindata.com





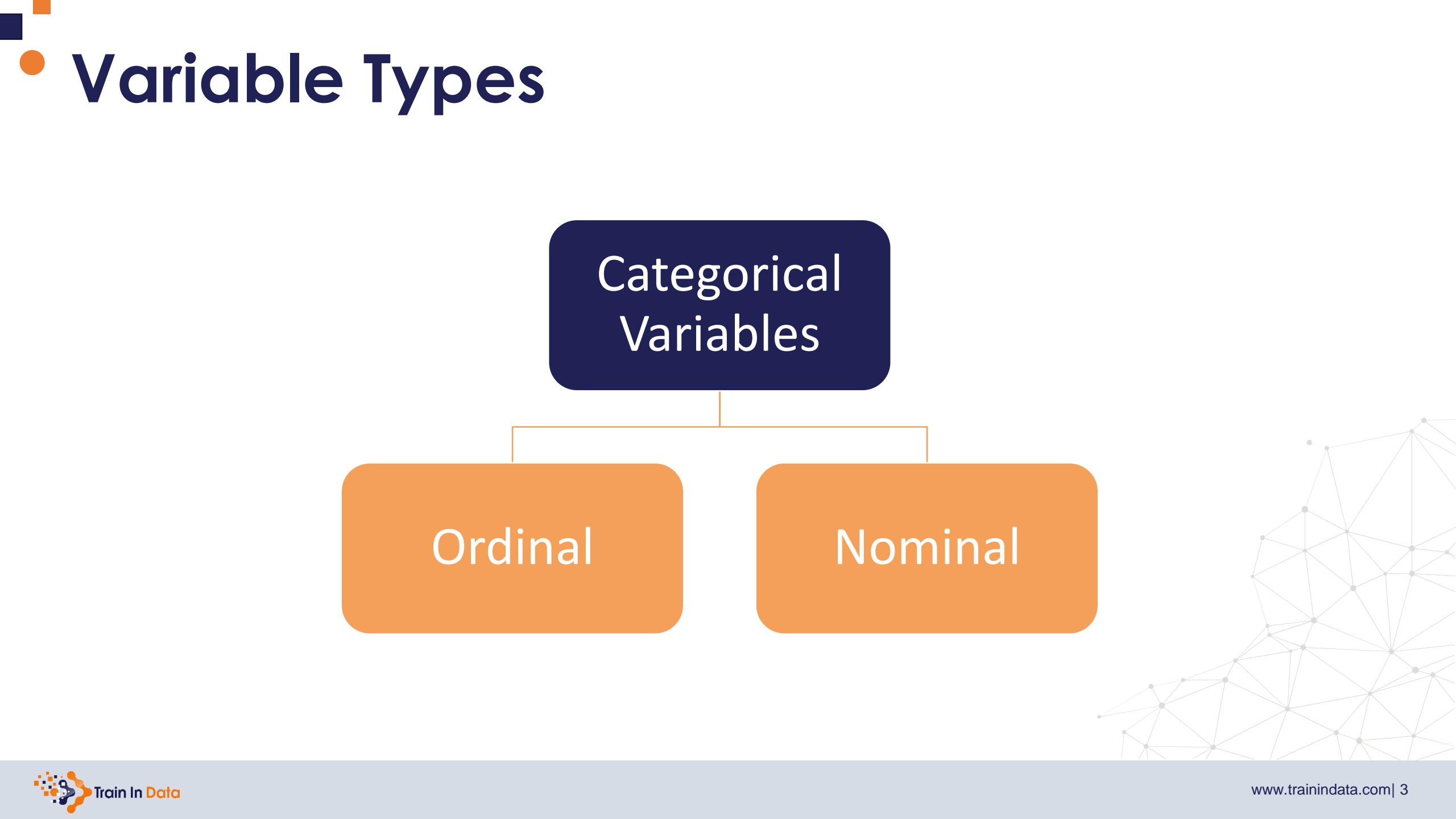
Categorical Variables

Categorical Variables

The values of a categorical variable are selected from a group of **categories**, also called **labels**. Examples:

- Marital status (married, single, ...)
- Intended use of loan (debt-consolidation, car purchase, ...)
- Mobile network provider (Vodafone, Orange, ...)
- Gender (male, female)





• Ordinal Variables

Categorical variables in which categories can be meaningfully ordered are called ordinal. Examples:

- Student's grade in an exam (A, B, C or Fail)
- Days of the week (Monday = 1 and Sunday = 7)
- Educational level, with the categories: Elementary school, High school, College graduate and PhD ranked from 1 to 4



Nominal Variables

Show no intrinsic order of the labels. Examples:

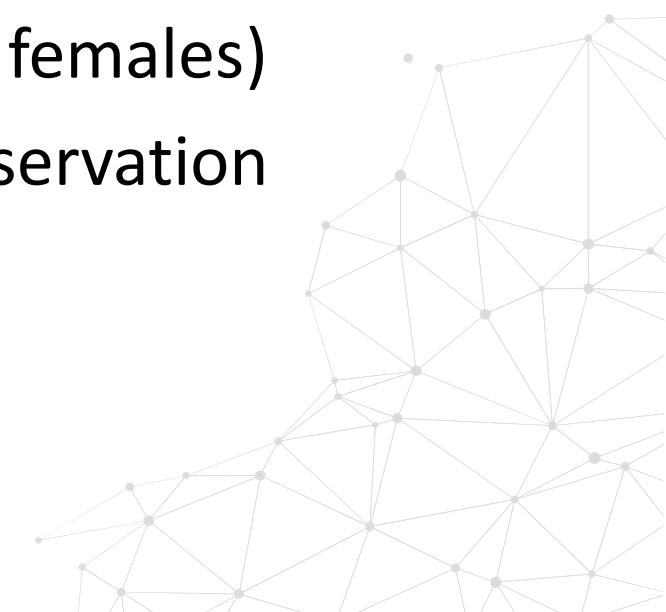
- Country of birth (Argentina, England, Germany)
- Postcode
- Vehicle make (Citroen, Peugeot, ...)



Special Cases

Special cases

- Categorical variables where categories are encoded as numbers (e.g. gender may be coded as 0 for males and 1 for females)
- Id variables: number that uniquely identifies an observation

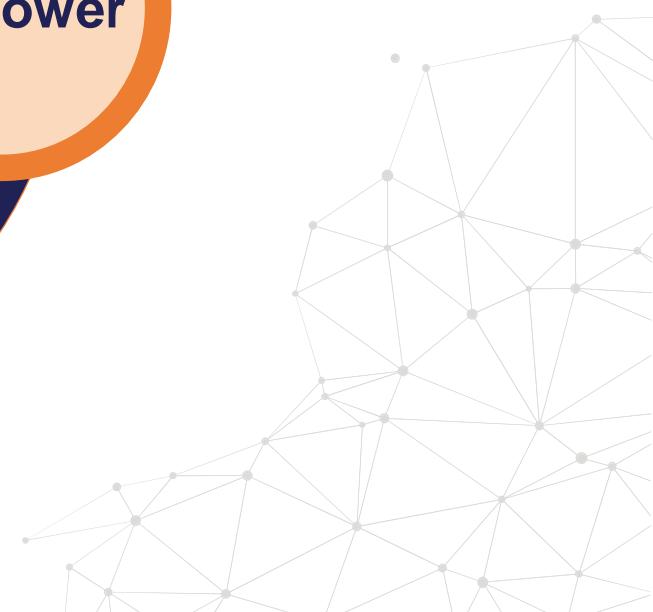
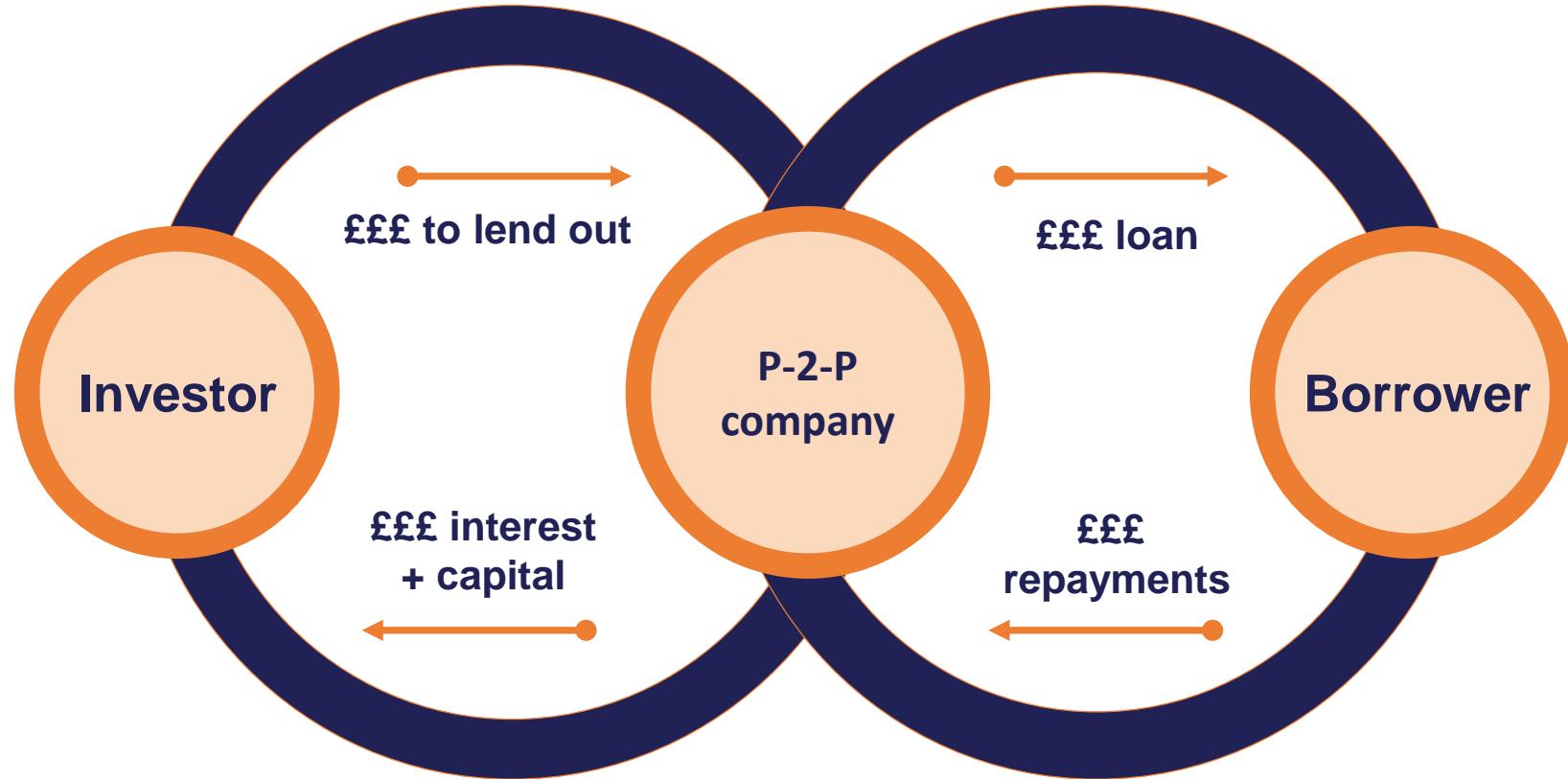


Categorical Variables

Notebook demo



Peer to Peer Finance



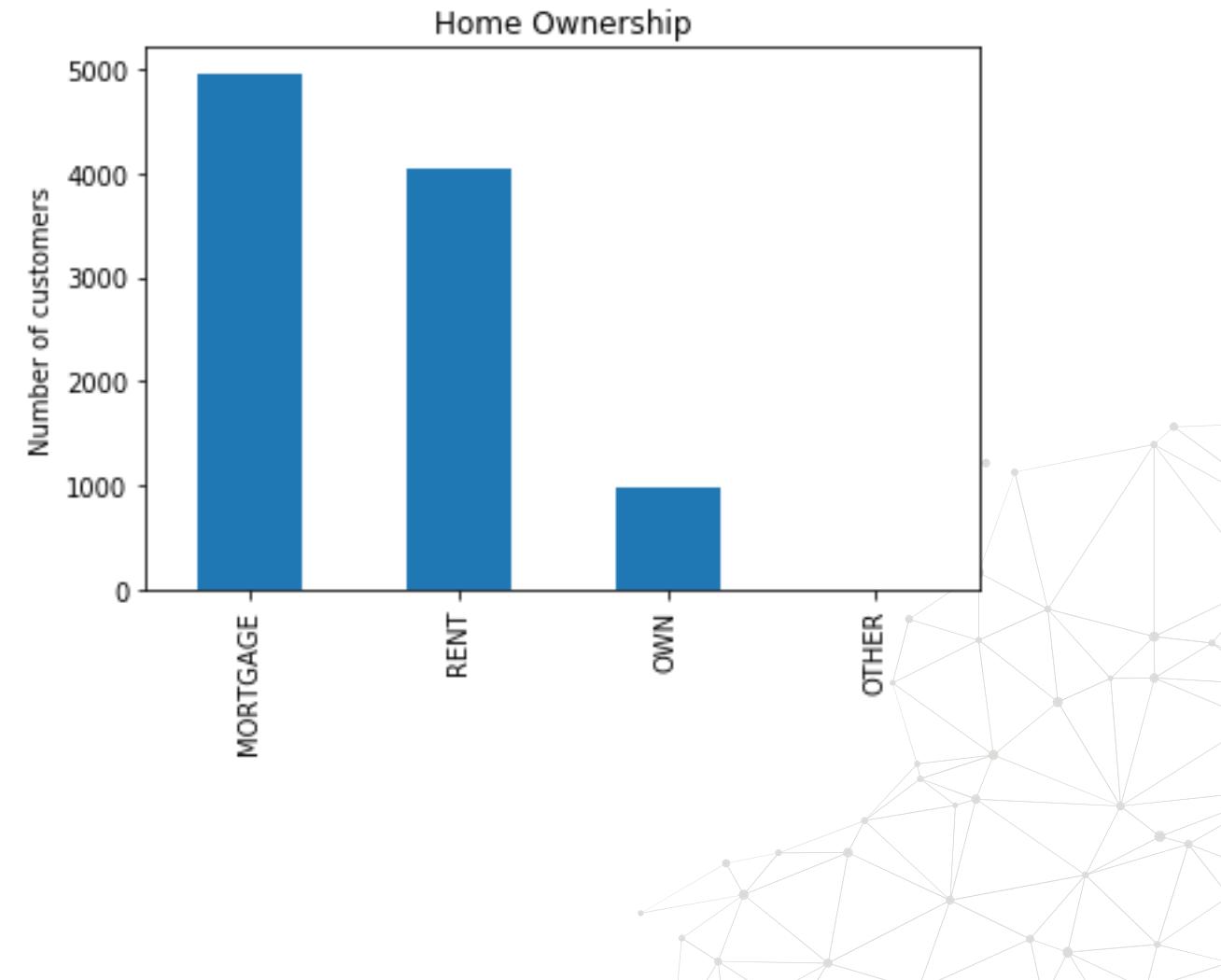
Categorical Variable Examples

- Intended use of loan - Nominal
- Home ownership - Nominal
- Loan status - Nominal



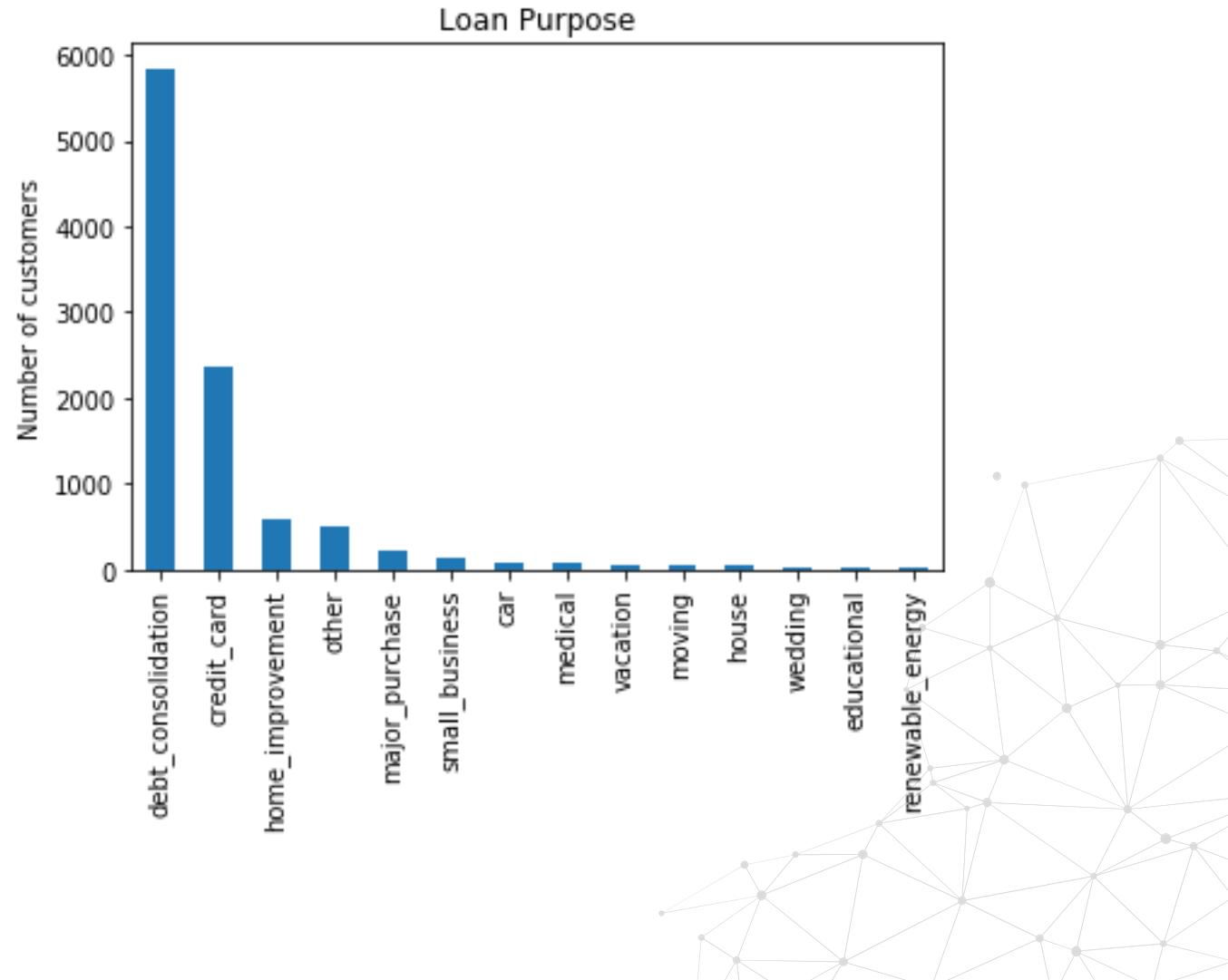
Home Ownership

- Example values: [Mortgage , Rent, Own, Other]



Loan Purpose

- Example values: [Debt consolidation, car, credit car, moving, etc.]



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook





THANK YOU

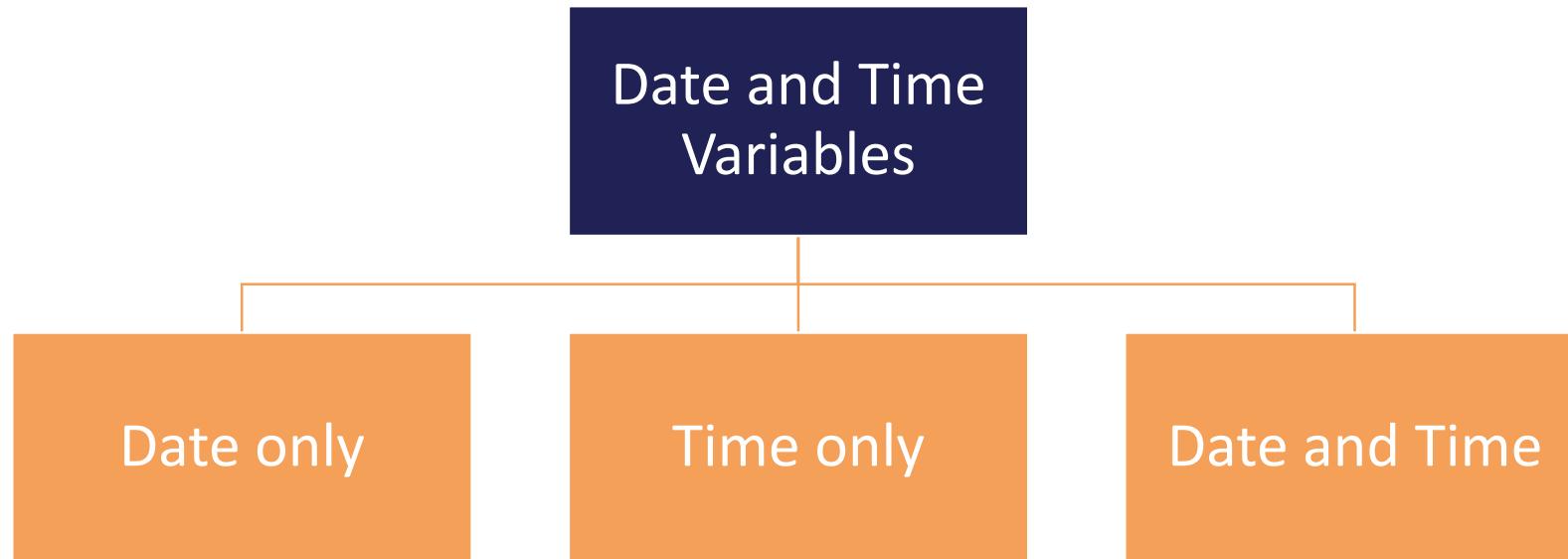
www.trainindata.com





Date and Time Variables

Date and Time Variables



Date and Time Variables

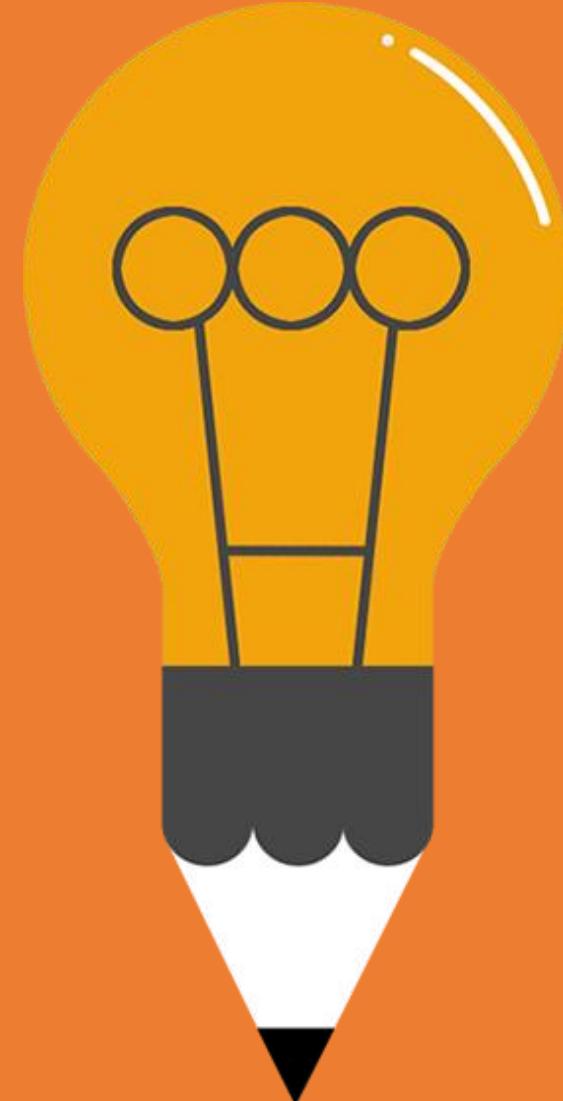
Date and time, or datetime variables they take dates and / or time as values.

- Date of birth ('29-08-1987', '12-01-2012')
- Date of application ('2016-Dec', '2013-March')
- Time of accident (12:20:45)
- Payment date ('29-08-1987 15:20.20')



Pre-processing of Date and Time

- We can enrich the dataset dramatically by extracting information from the date and time.
- We will see how in a dedicated section.

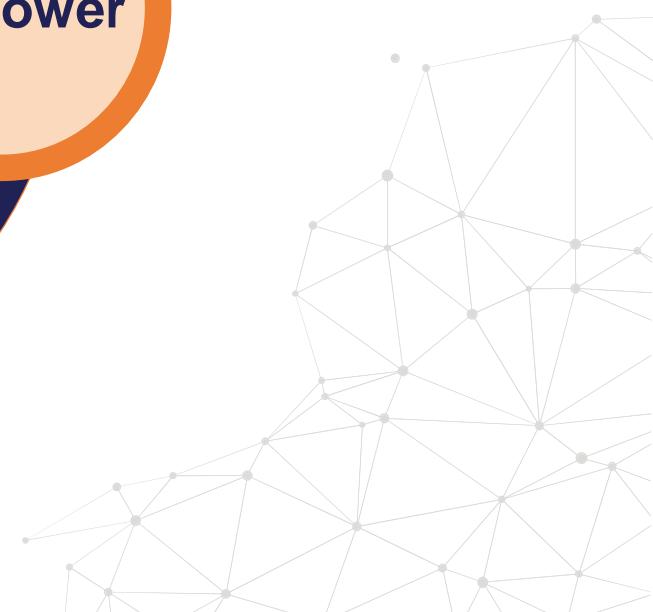
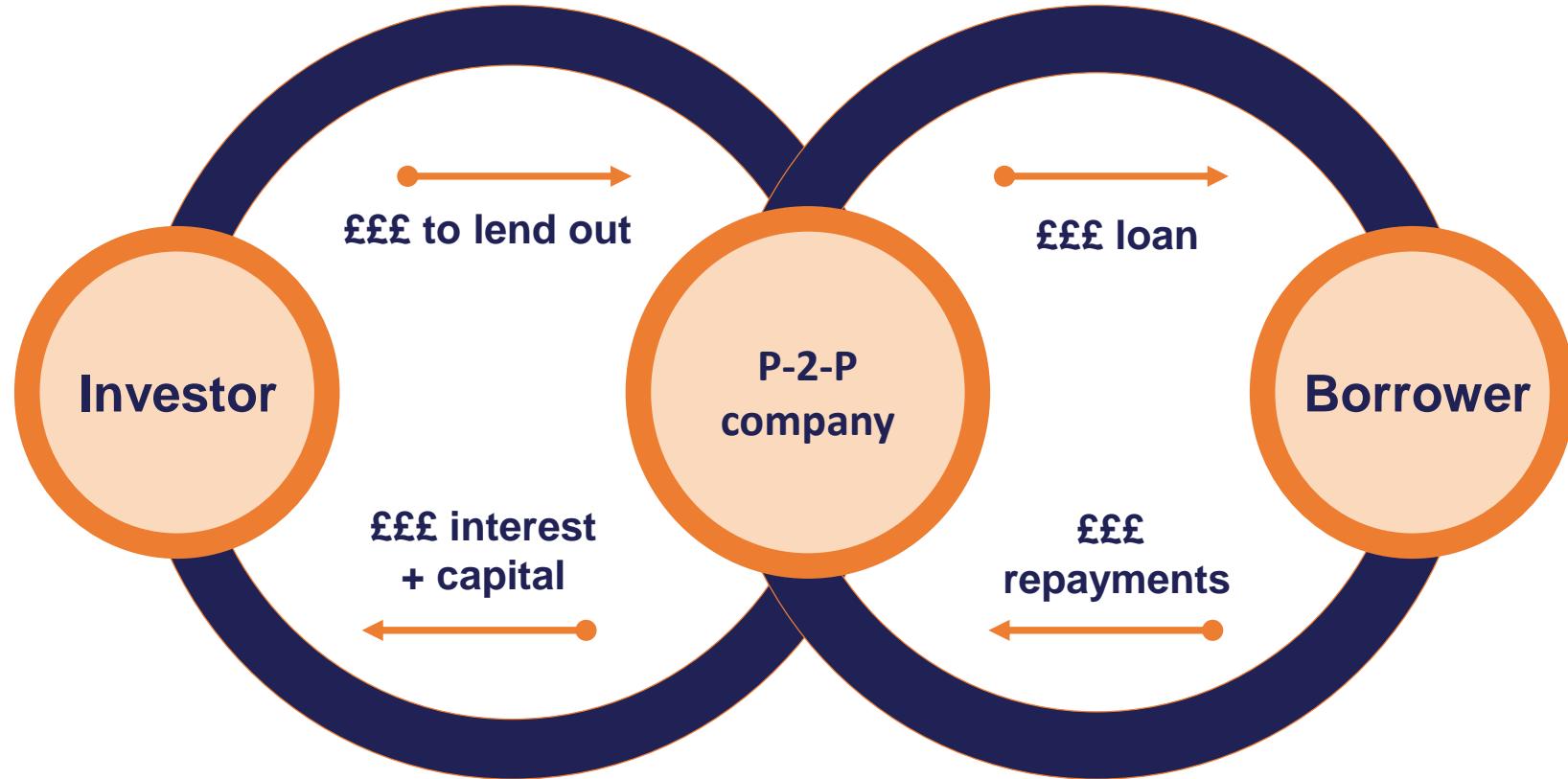


Date and Time Variables

Notebook demo



Peer to Peer Finance



Date Variable Examples

- Loan issued date
- Last repayment date

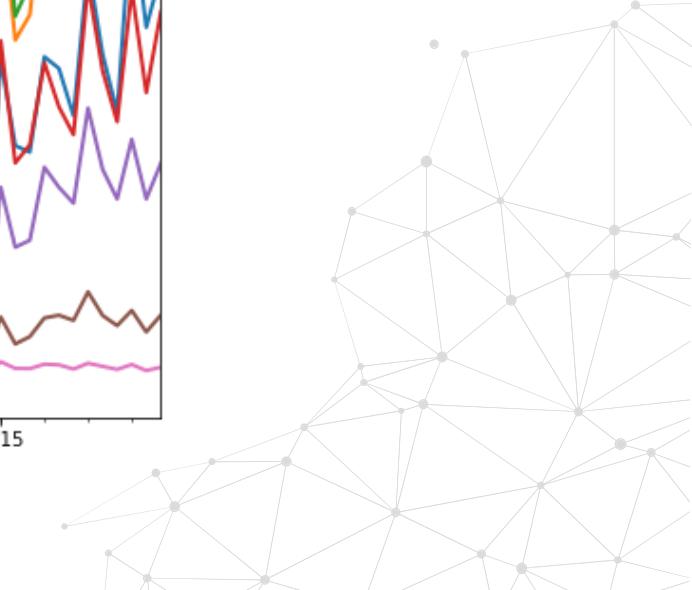
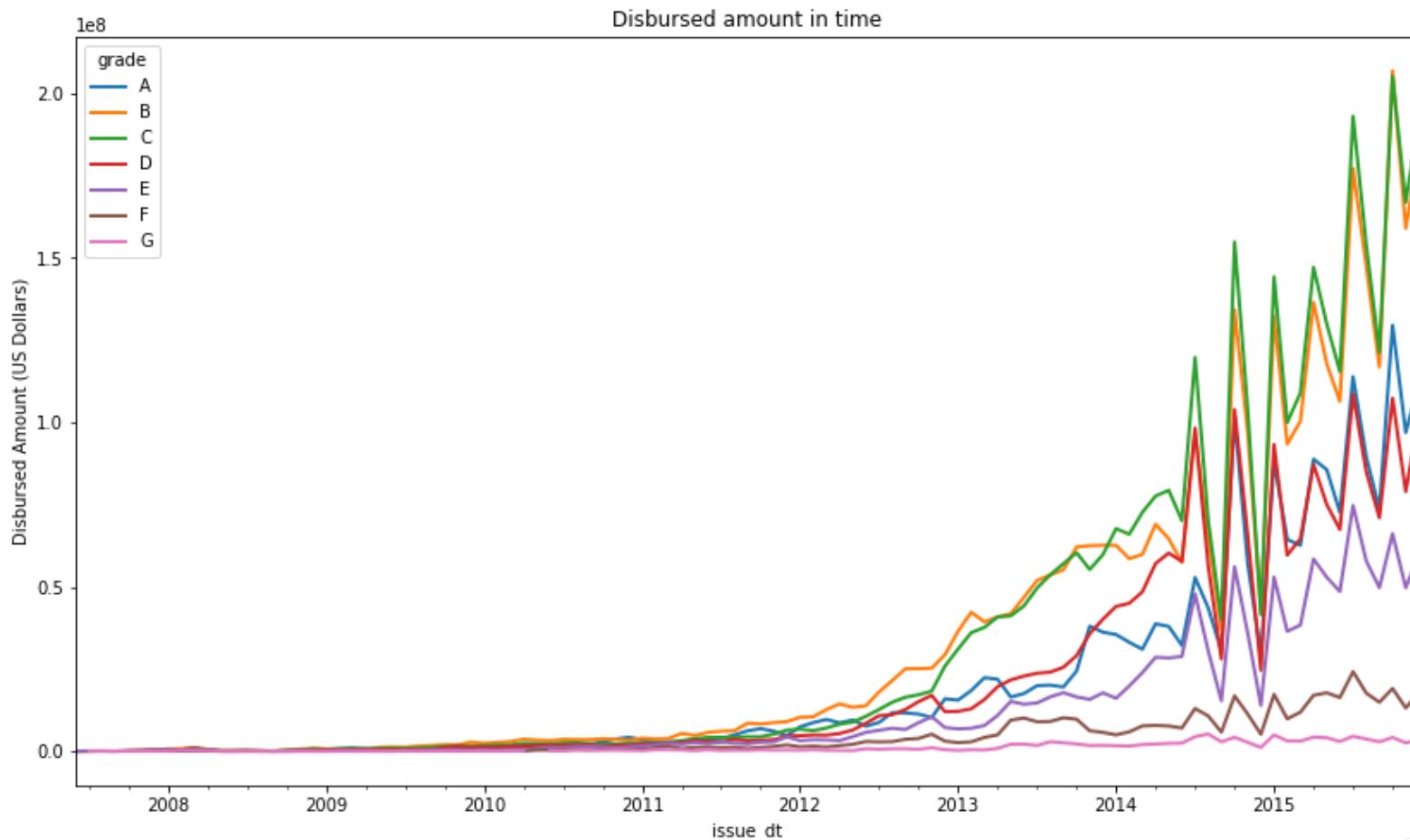


Date Variable Examples

	loan_amnt	grade	issue_d	purpose	last_pymnt_d
0	5000.0	B	Dec-2011	credit_card	Jan-2015
1	2500.0	C	Dec-2011	car	Apr-2013
2	2400.0	C	Dec-2011	small_business	Jun-2014
3	10000.0	C	Dec-2011	other	Jan-2015
4	3000.0	B	Dec-2011	other	Jan-2016



Issued Date



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook





THANK YOU

www.trainindata.com

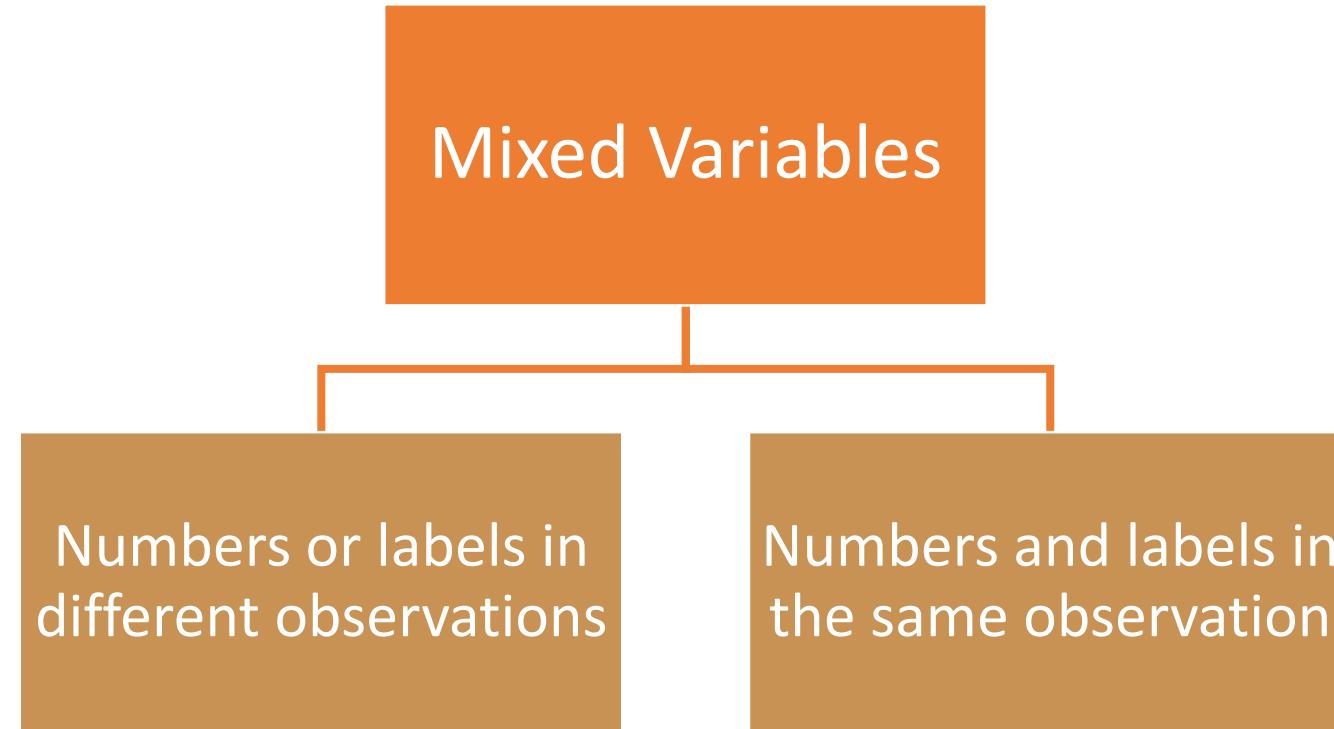




Train In Data

Mixed Variables

Mixed Variables



Mixed Variables

Observations show either numbers or categories among their values

- Number of credit accounts (1-100, U, T, M)
 - U = unknown, T = unverified, M = unmatched)
- Number of missed payments (1-3, D, A)
 - D = defaulted, A = arrangements



Mixed Variables

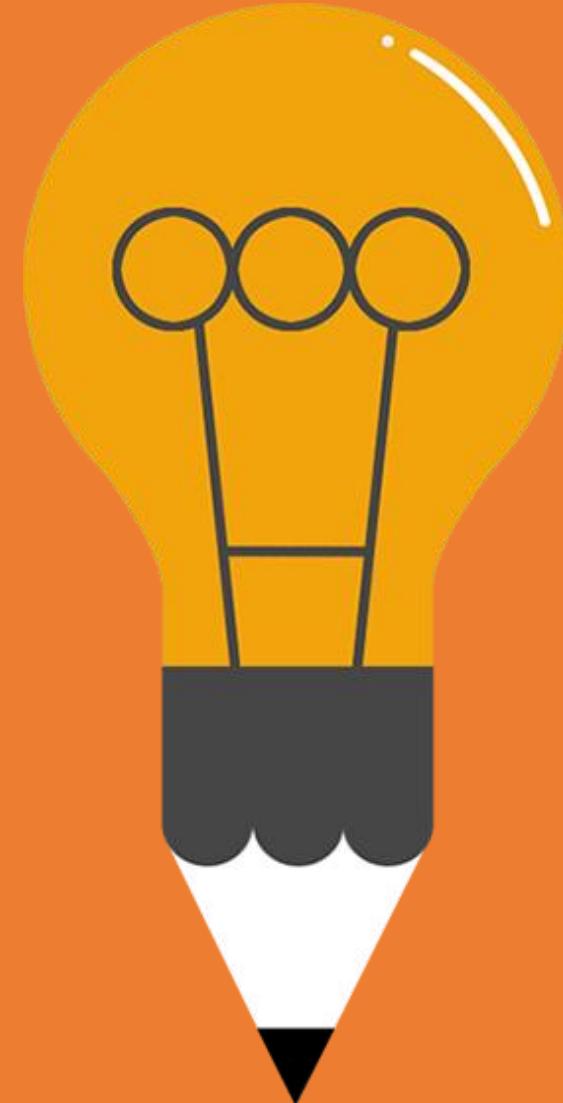
Observations show both numbers and categories in their values

- Cabin (Titanic) (A15, B18, ...)
- Ticket (Titanic) (A103349)
- Vehicle registration (AB500)
- Postcode (SE18)



Pre-processing mixed variables

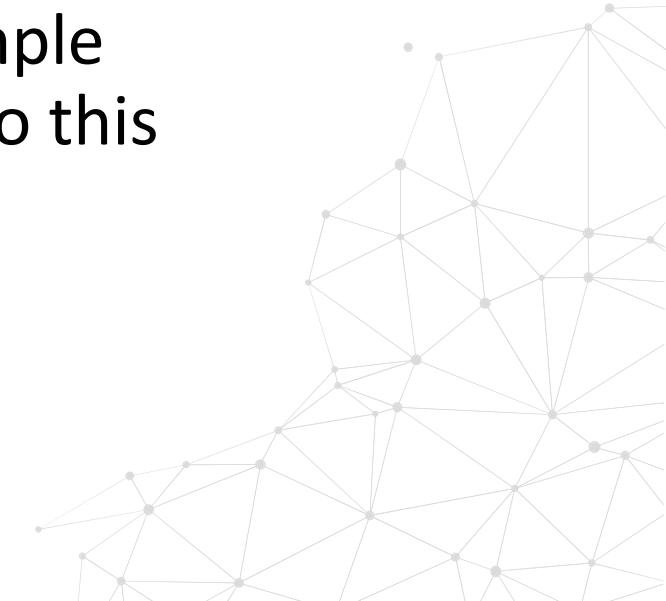
- We can enrich the dataset dramatically by extracting information from the label and the number of the mixed variable.
- We will learn how in a dedicated section.



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
- Download the sample dataset attached to this lecture.





THANK YOU

www.trainindata.com





Variable Characteristics

Objectives

Understand the characteristics of the variables which need to be addressed before building machine learning models. Specifically:

- Identify variable characteristics
- Understand how they impact machine learning models
- Examples of variables characteristics in real datasets



Content



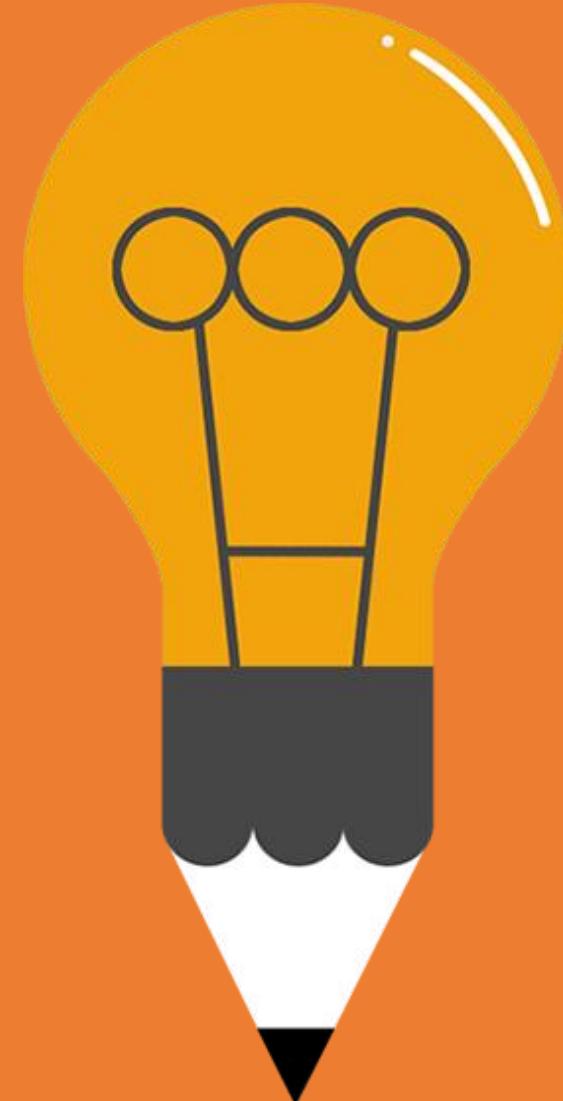
For each lecture:

- Presentation and video
- Accompanying Jupyter notebook
 - Examples of the variable characteristics in real datasets
 - Code to identify and the different variable characteristics



Final Summary

- Final article summarizing how the different variable characteristics affect the different machine learning models at the end of the section.
- Additional reading resources.



▪ Variable ▪ Characteristics

Understand what are the different things we need to look out for when analyzing the variables in our datasets



Variable Characteristics





THANK YOU

www.trainindata.com





Missing Data

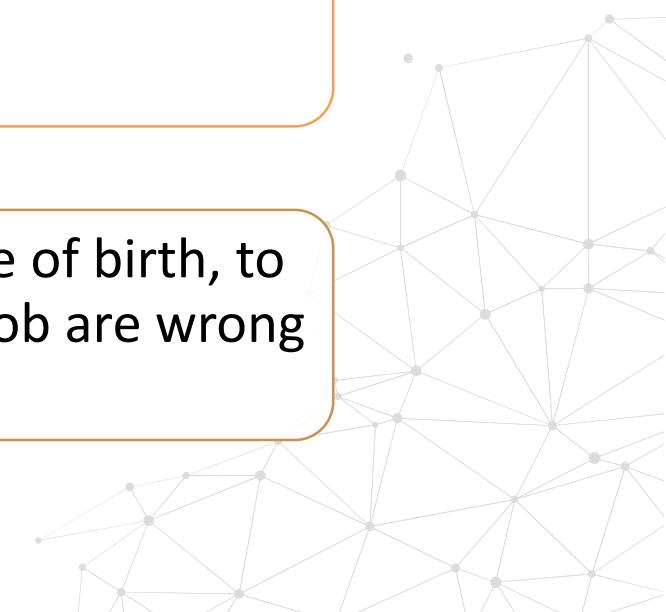
Missing Data: Definition

- Missing data, or missing values, occur when no data is stored for a certain observation in a variable.
- Missing data are a common occurrence in most datasets
- Missing data can have a significant effect on the conclusions that can be drawn from the data.

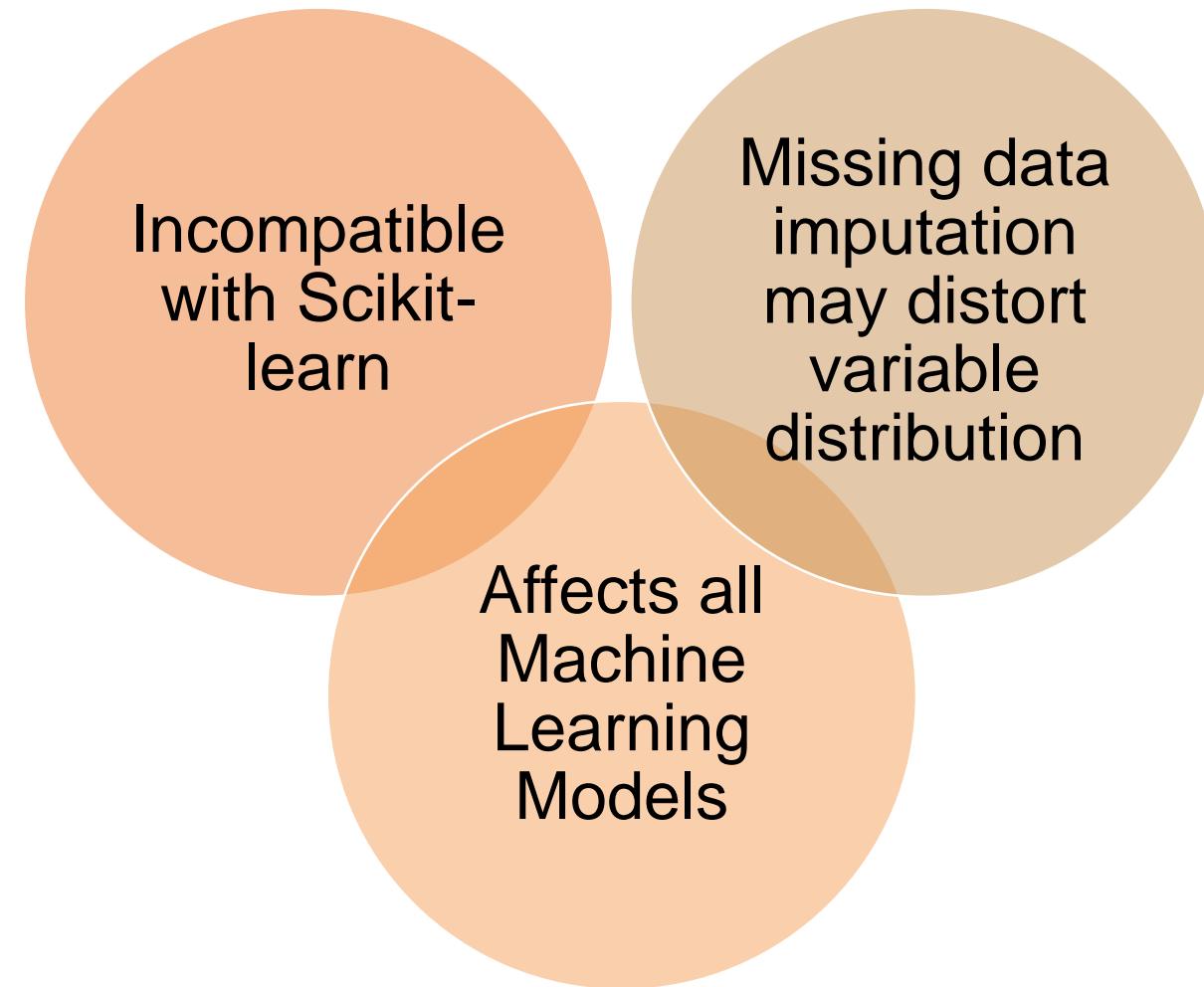


Missing Data: Causes

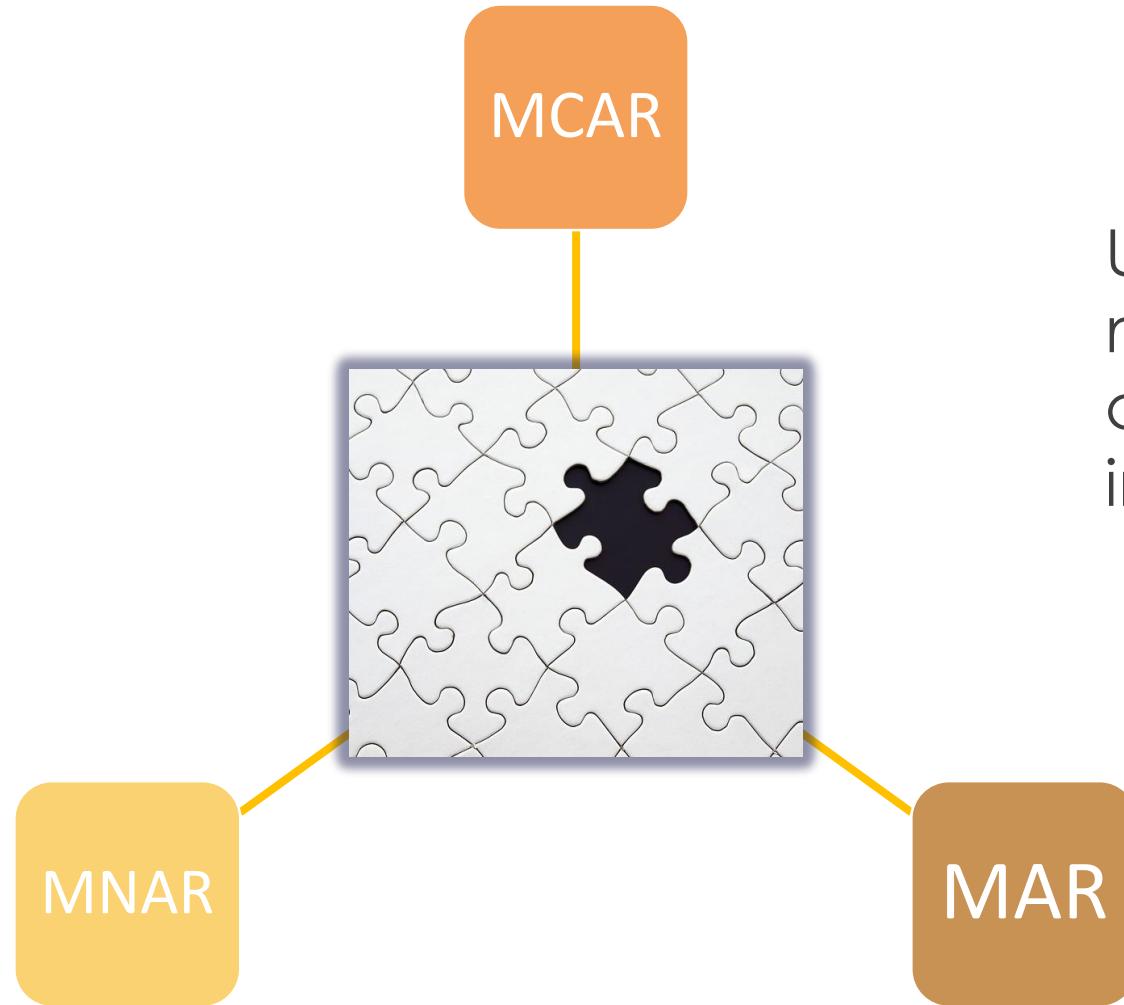
- A value is missing because it was forgotten, lost or not stored properly.
Lost
- E.g., a variable is created from the division of 2 variables and the denominator takes 0.
Don't exist
- E.g., when matching data against postcode, or date of birth, to enrich with more variables, and the postcode or dob are wrong or don't exist, the new variables will take NA.
Not found |
Not Identified



Missing Data: Impacts



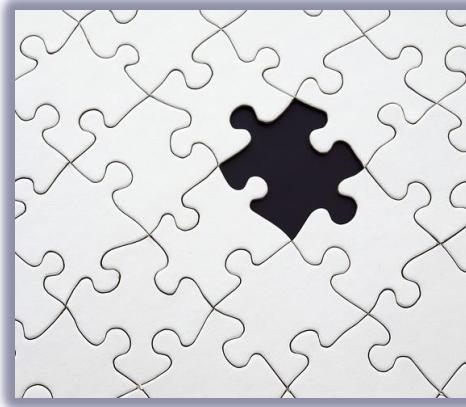
Missing Data: Mechanisms



Understanding the missing data mechanisms may help us choose the right missing data imputation technique



• Missing Data Completely at Random (MCAR)



- The probability of being missing is the same for all the observations
- There is absolutely no relationship between the data missing and any other values, observed or missing, within the dataset
- Disregarding those cases would not bias the inferences made



Image taken from [here](#)

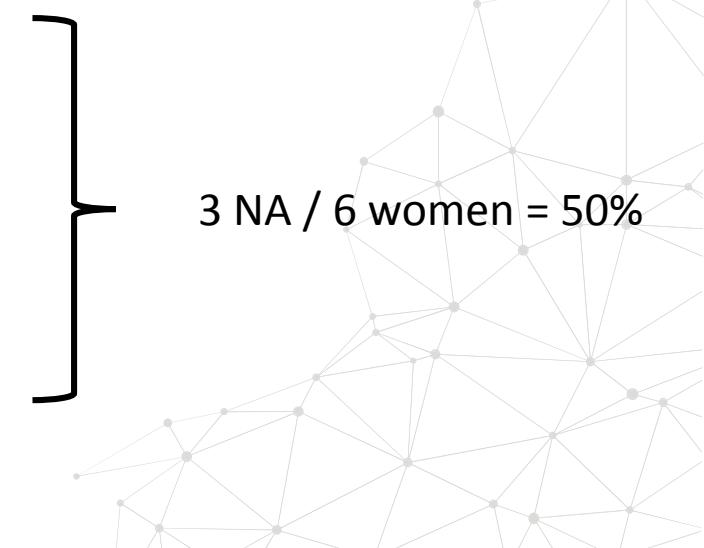
Missing Data at Random (MAR)

- The probability of an observation being missing depends on available information

Gender	Weight
Male	60 kg
Male	NA
Male	NA
Male	77 kg
Male	80 kg
Male	62 kg
Female	NA
Female	NA
Female	60 kg
Female	55 kg
Female	NA
Female	58 kg

2 NA / 6 men = 33%

3 NA / 6 women = 50%



Missing Data not at Random (MNAR)

- there is a mechanism or a reason why missing values are introduced in the dataset.

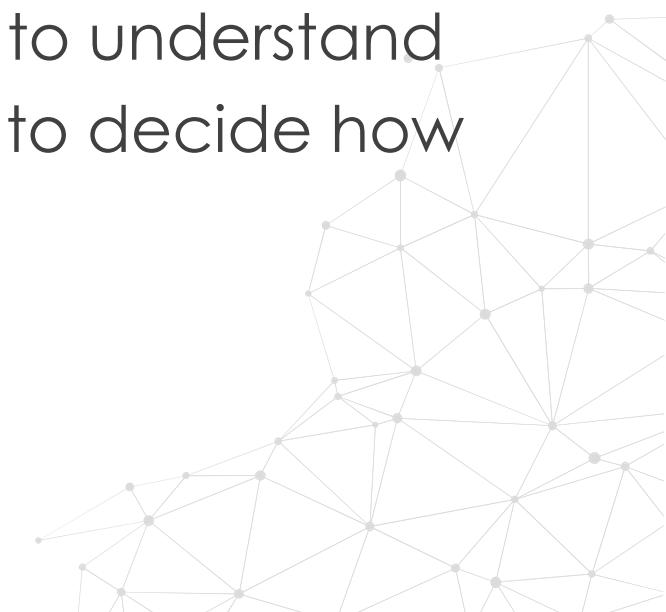
Target = depression	No of clinic visits	No sports classes weekly
Yes	1	NA
Yes	NA	NA
Yes	NA	0
Yes	4	2
Yes	NA	1
Yes	3	NA
No	0	0
No	NA	5
No	1	2
No	1	1
No	2	1
No	NA	2

More NA overall for depressed patients

Less NA for non-depressed patients

In addition

- To understand the mechanisms by which missing data is introduced, we need to become familiar with the methods used for data collection.
- This is not always possible. However, it is a good idea to understand the methods of data collection as much as possible, to decide how best to engineer the features.



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
- Examples of MCAR, MAR and MNAR
- Titanic dataset
- Loan Book from P-2-P company





THANK YOU

www.trainindata.com





Cardinality

• Cardinality definition

- The values of a categorical variable are selected from a group of categories (also called labels).
- The number of different labels is known as **cardinality**.

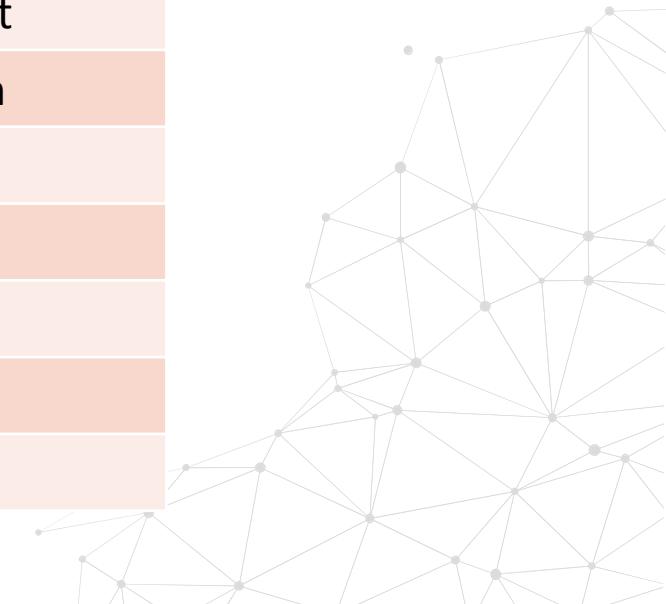


• Cardinality examples

- The variable gender contains only 2 labels in this example
- Vehicle Make contains 9 labels in the example table
- The variables city or postcode, can contain a huge number of different labels.

Gender	Vehicle Make
Male	Mercedes
Male	Ford
Male	Ford
Male	Renault
Male	Seat
Male	Renault
Female	Citroen
Female	Toyota
Female	Kia
Female	Kia
Female	Nissan
Female	BMW

Gender → 2
Vehicle Make → 9





Cardinality effects

Are multiple labels in a categorical variable a problem?



• Cardinality: Impacts

Strings are incompatible with Scikit-Learn

Uneven distribution between train and test sets

Over-fitting in tree based algorithms

Operational problems



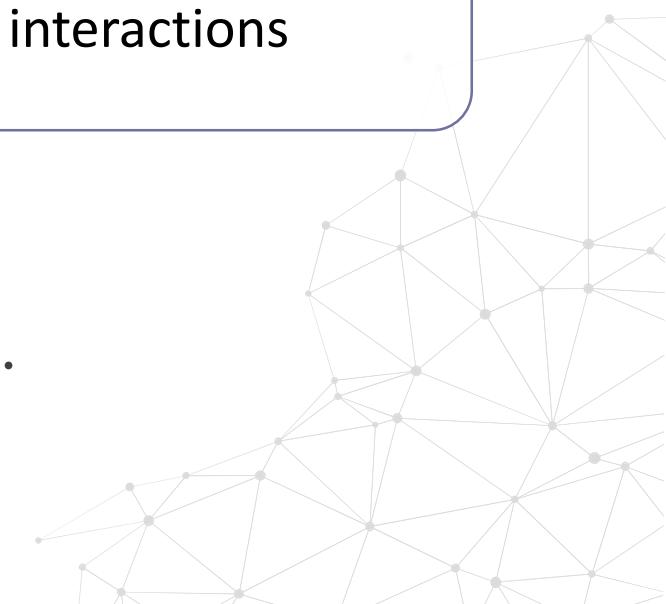
• Strings and categorical encoding

Scikit-Learn does
not support
strings as inputs

Categories must
be encoded as
numbers

Encoding techniques
impact feature space
and variable
interactions

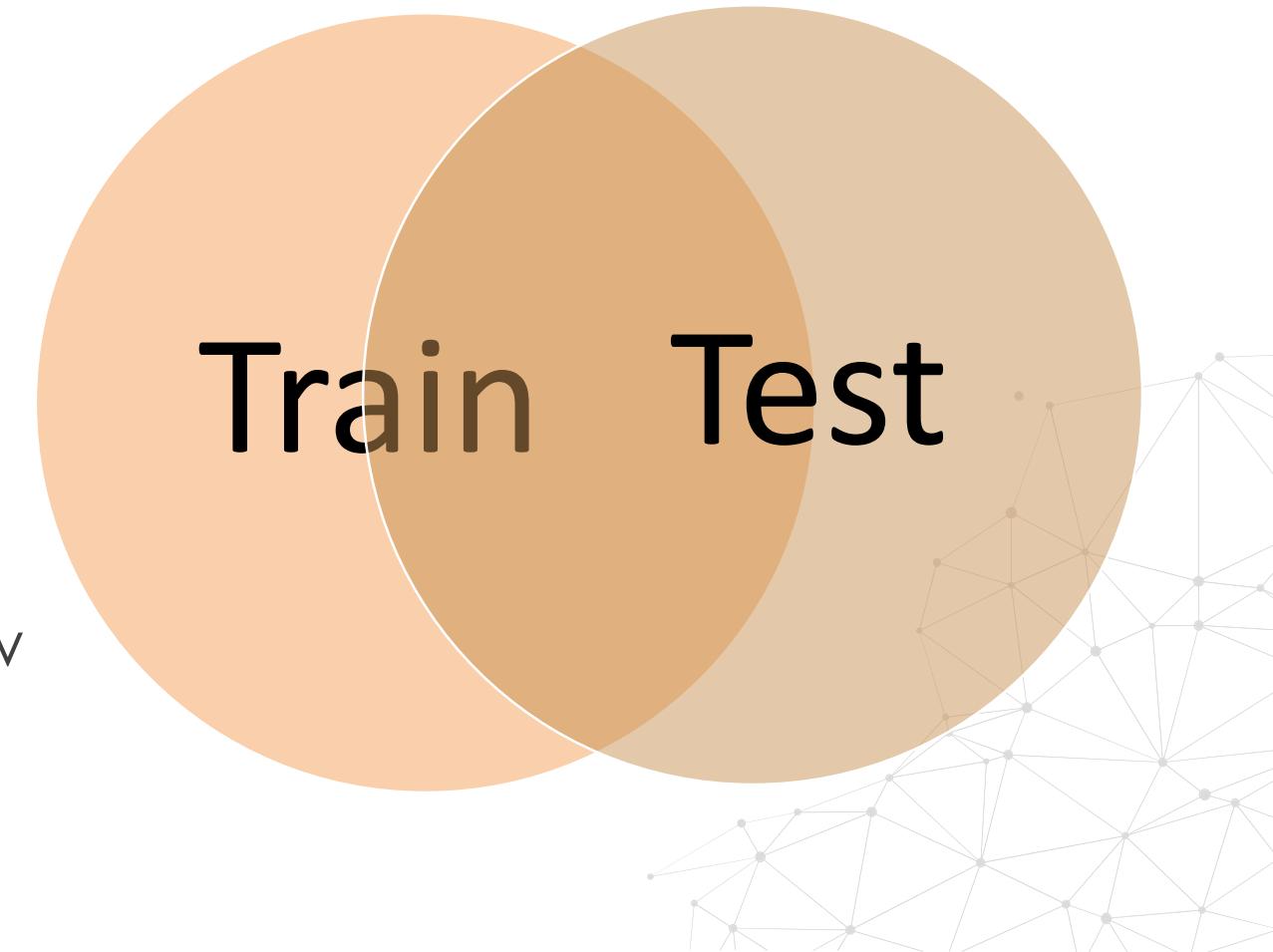
More on encoding methods in a dedicated section...



• Uneven distribution between train and test sets

For highly cardinal variables:

- Some labels may appear only in train set → over-fitting
- Some labels may appear only in test set → model will not know how to interpret the values



• Uneven distribution

Obs	Vehicle Make
1	Mercedes
2	Ford
3	Ford
4	Renault
5	Seat
6	Renault
7	Citroen
8	Toyota
9	Kia
10	Kia
11	Nissan
12	BMW



Obs	Vehicle Make
1	Mercedes
3	Ford
6	Renault
7	Citroen
9	Kia
11	Nissan

Train Set



Obs	Vehicle Make
2	Ford
5	Seat
4	Renault
8	Toyota
10	Kia
12	BMW

Test Set

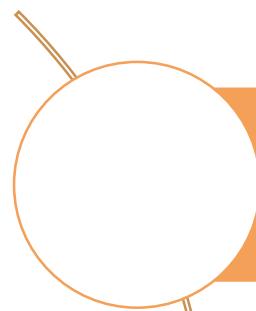




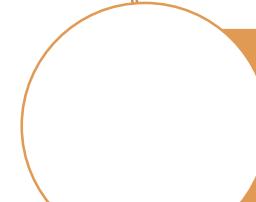
Overfitting



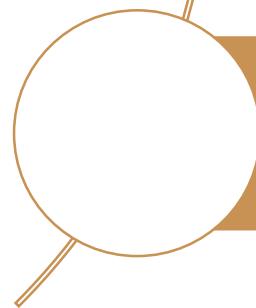
• Cardinality and overfitting



Variables with too many labels tend to dominate over those with fewer labels, particularly in **tree based algorithms**.



A big number of labels within a variable may introduce noise with little, if any, information



Reducing cardinality may help improve model performance

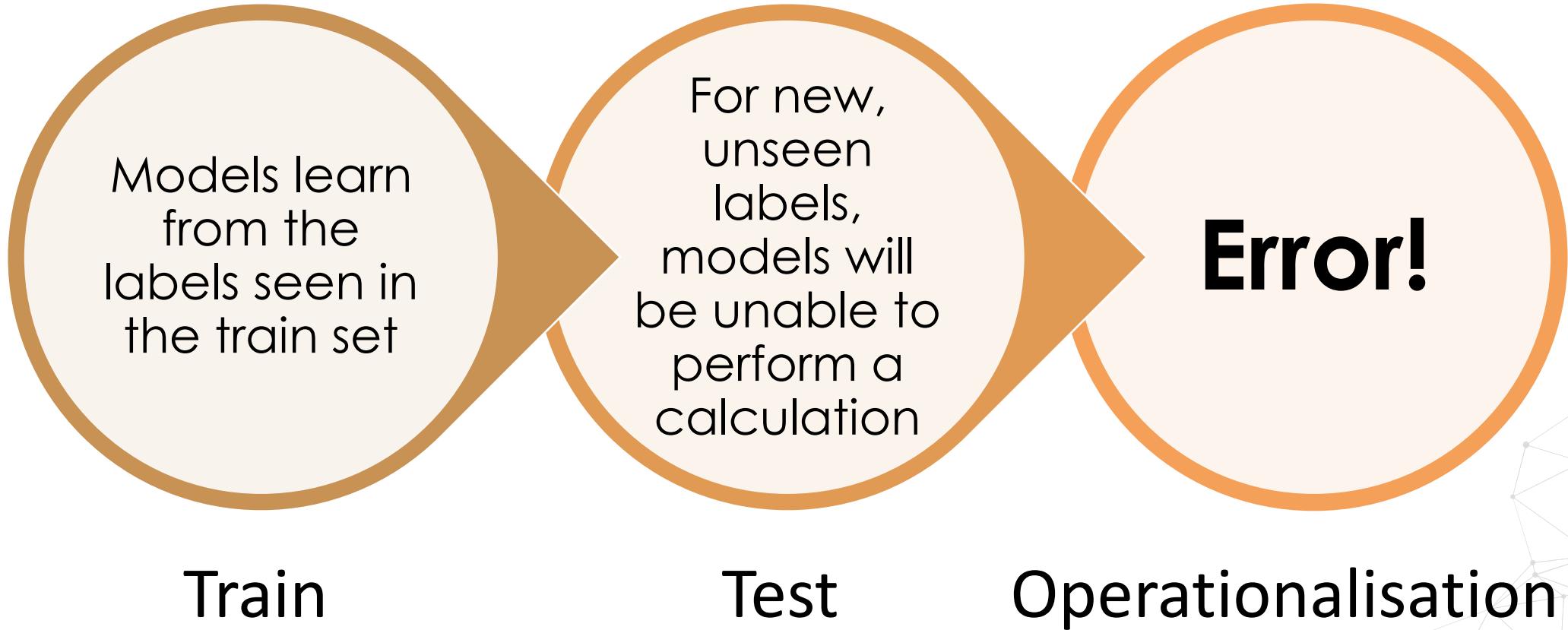


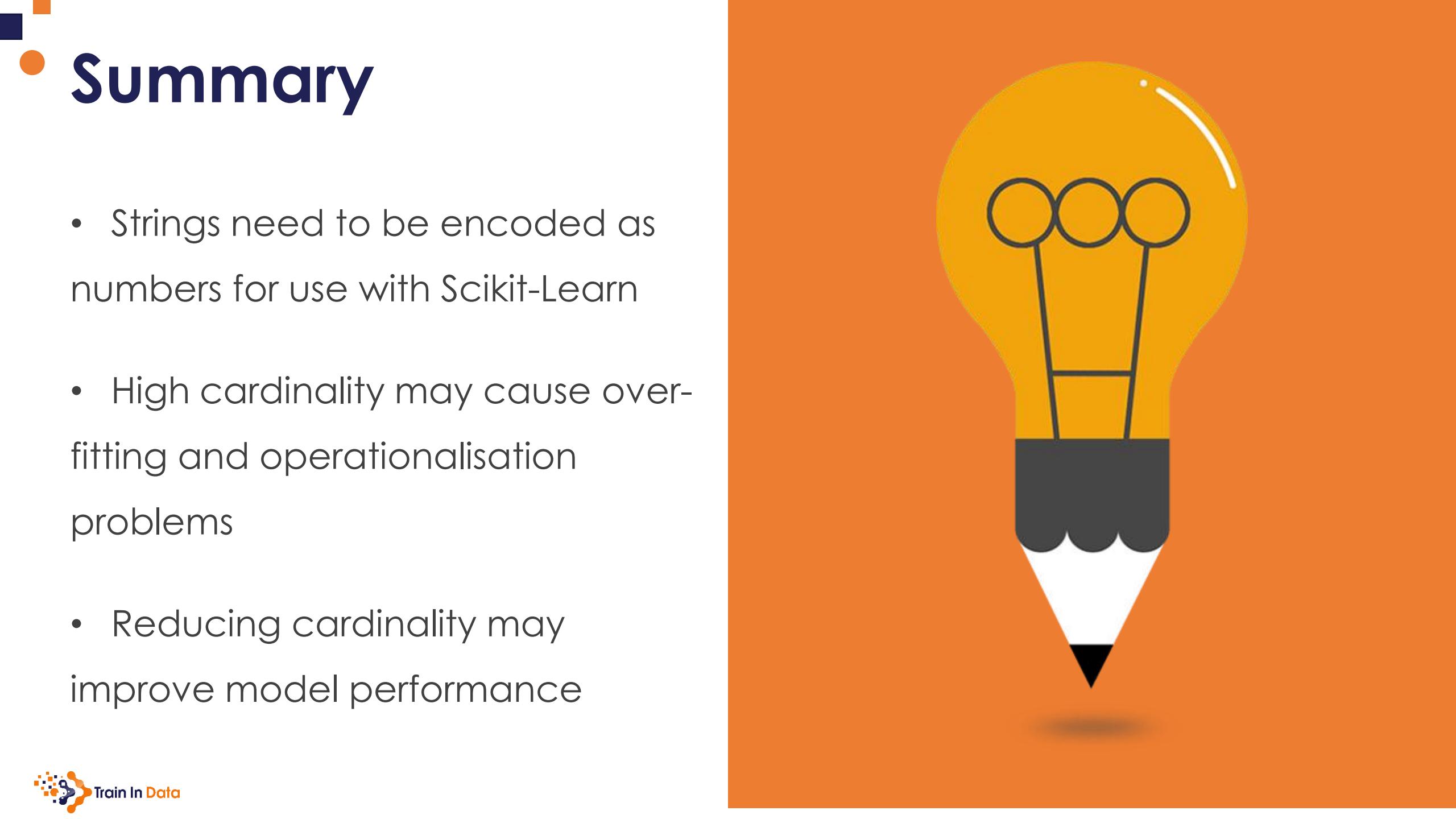


Operational problems



• Cardinality and operationalisation





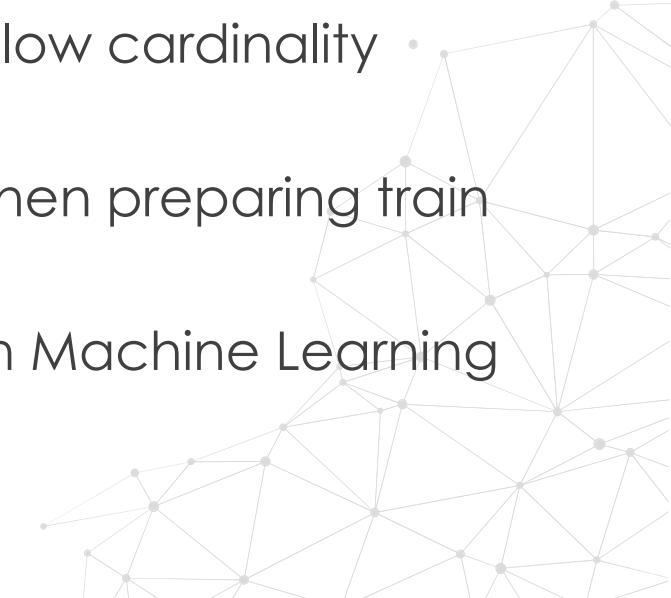
Summary

- Strings need to be encoded as numbers for use with Scikit-Learn
- High cardinality may cause overfitting and operationalisation problems
- Reducing cardinality may improve model performance

• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
- How to quantify cardinality
- Examples of high and low cardinality variables
- Effect of cardinality when preparing train and test sets
- Effect of cardinality on Machine Learning Model performance





THANK YOU

www.trainindata.com

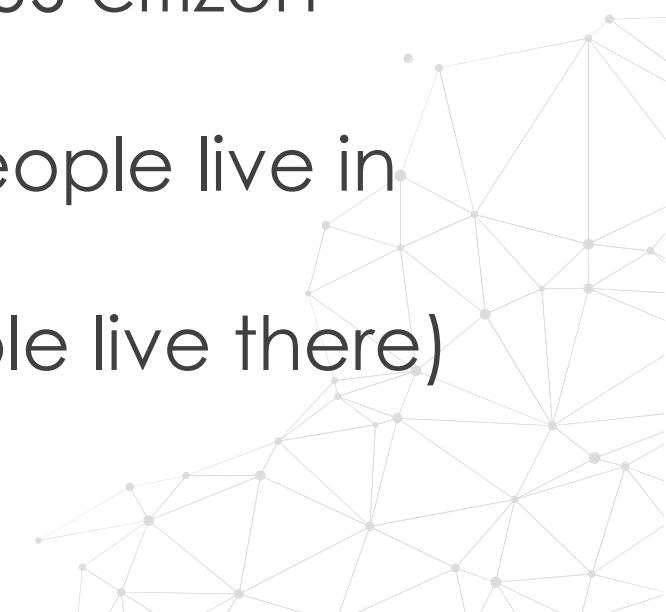




Rare Labels

Rare Labels

- Rare labels are those that appear only in a tiny proportion of the observations in a dataset
- For example, for the variable “city” where a US citizen lives:
 - New York is a frequent category (many people live in New York)
 - Leavenworth is a rare category (few people live there)





Rare Labels effects

How do Rare labels affect the performance and operationalization of Machine Learning Models?



The background of the slide features a blurred, monochromatic image of a city skyline at night. The buildings are tall and closely packed, with numerous windows glowing with light. In the foreground, the dark, rippling surface of a body of water is visible, with some distant lights reflecting on its surface.

“Same impacts and considerations
as with high cardinality ”

- # Rare labels impacts

Strings are incompatible with Scikit-Learn

Uneven distribution between train and test sets

Over-fitting in tree based algorithms

Operational problems



Rare label example

Obs	Gender	Vehicle Make
1	Male	Mercedes
2	Male	Ford
3	Male	Ford
4	Male	Renault
5	Male	Seat
6	Male	Renault
7	Female	Citroen
8	Female	Toyota
9	Female	Kia
10	Female	Kia
11	Female	Nissan
12	Female	BMW

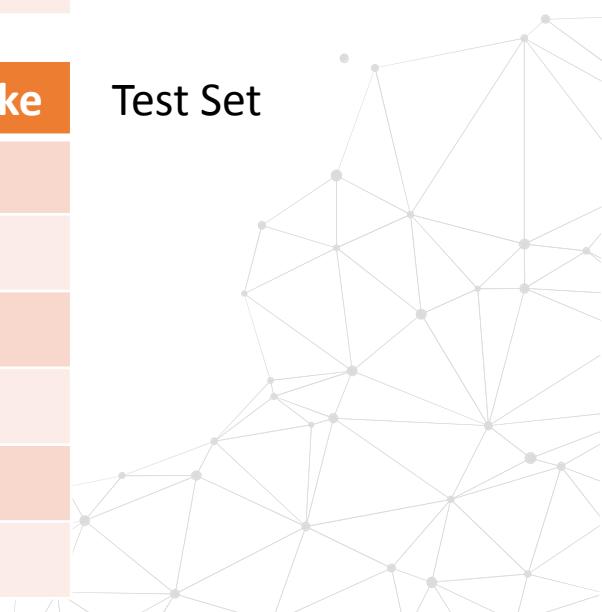


Obs	Gender	Vehicle Make
1	Male	Mercedes
3	Male	Ford
6	Male	Renault
7	Female	Citroen
9	Female	Kia
11	Female	Nissan

Train Set

Obs	Gender	Vehicle Make
2	Male	Ford
5	Male	Seat
4	Male	Renault
8	Female	Toyota
10	Female	Kia
12	Female	BMW

Test Set



Rare label example

Obs	Gender	Vehicle Make
1	Male	Mercedes
2	Male	Ford
3	Male	Ford
4	Male	Renault
5	Male	Seat
6	Male	Renault
7	Female	Citroen
8	Female	Toyota
9	Female	Kia
10	Female	Kia
11	Female	Nissan
12	Female	BMW

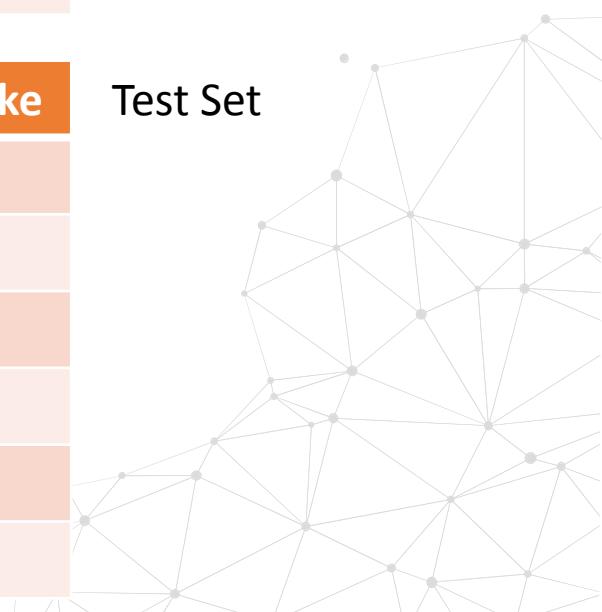


Obs	Gender	Vehicle Make
1	Male	Mercedes
3	Male	Ford
6	Male	Renault
7	Female	Citroen
9	Female	Kia
11	Female	Nissan

Train Set

Obs	Gender	Vehicle Make
2	Male	Ford
5	Male	Seat
4	Male	Renault
8	Female	Toyota
10	Female	Kia
12	Female	BMW

Test Set



Rare label example

Obs	Gender	Vehicle Make
1	Male	Mercedes
2	Male	Ford
3	Male	Ford
4	Male	Renault
5	Male	Seat
6	Male	Renault
7	Female	Citroen
8	Female	Toyota
9	Female	Kia
10	Female	Kia
11	Female	Nissan
12	Female	BMW



Obs	Gender	Vehicle Make
1	Male	Mercedes
3	Male	Ford
6	Male	Renault
7	Female	Citroen
9	Female	Kia
11	Female	Nissan

Obs	Gender	Vehicle Make
2	Male	Ford
5	Male	Seat
4	Male	Renault
8	Female	Toyota
10	Female	Kia
12	Female	BMW

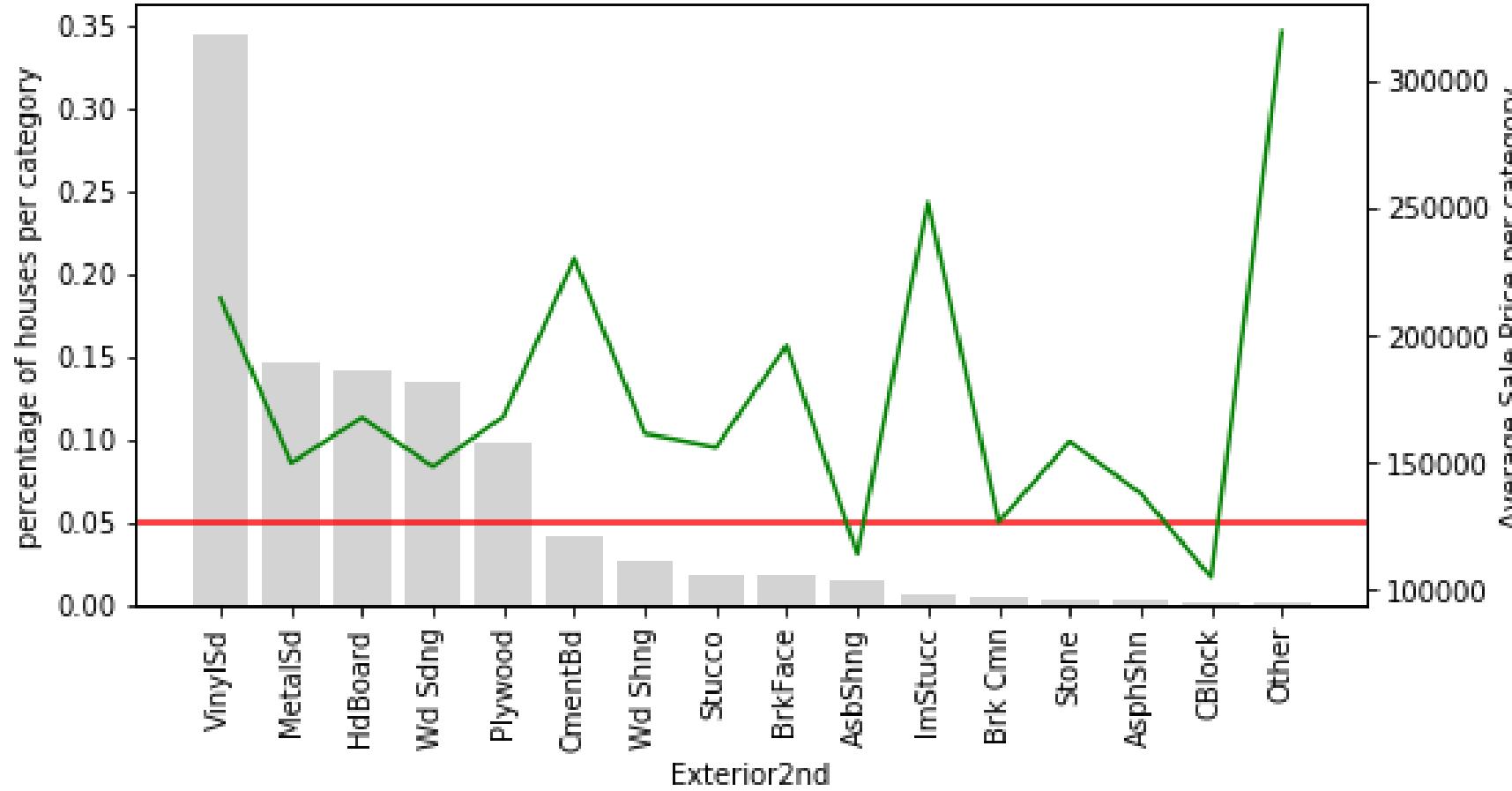
Train Set

Potential Overfit

Test Set

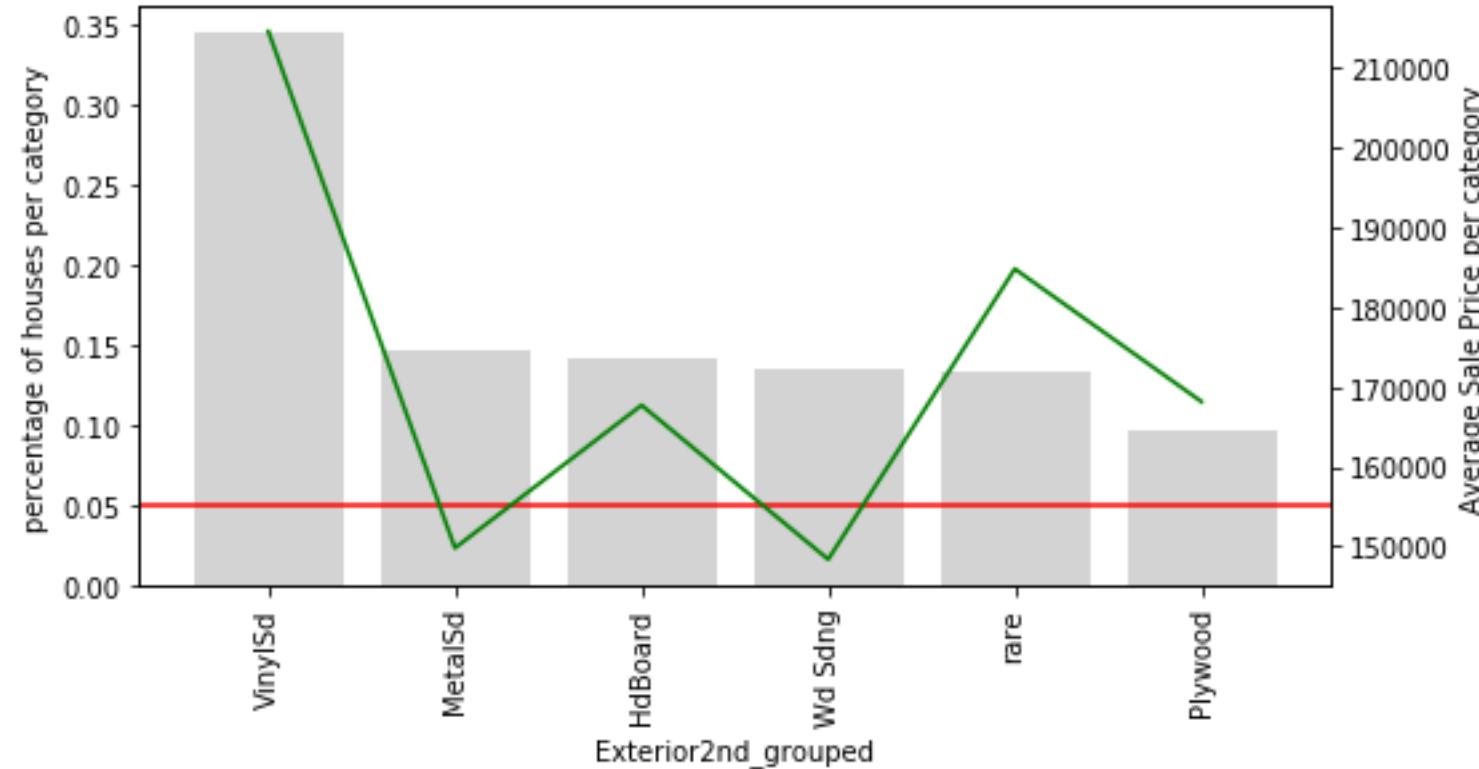
Potential Operation alisation problem

Rare labels and deriving information



Hard to understand the true effect of the rare label on the outcome

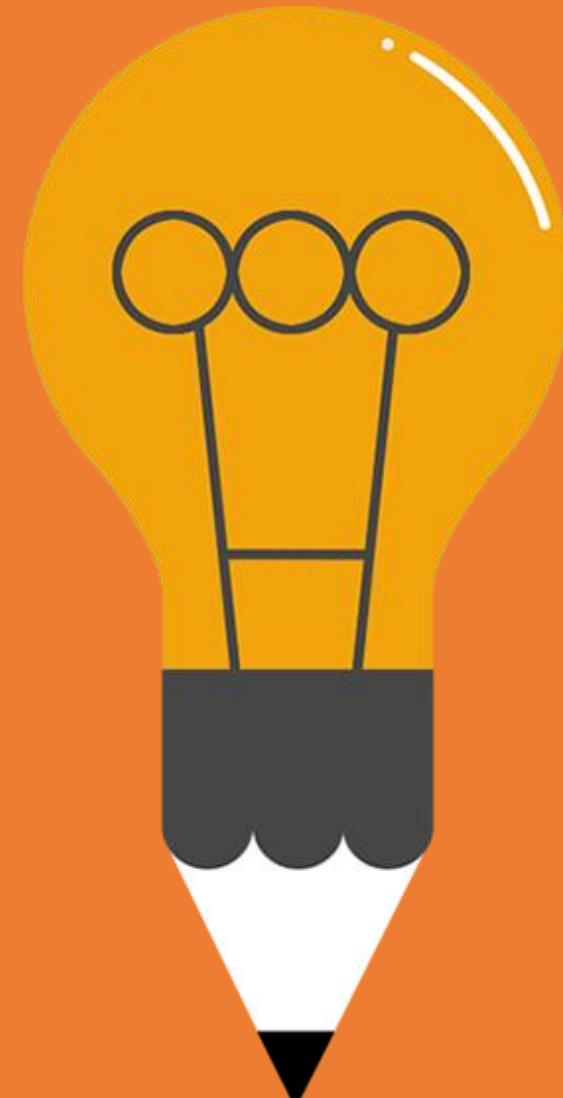
Rare labels and deriving information



Hard to understand the true effect of the rare label on the outcome

• Summary

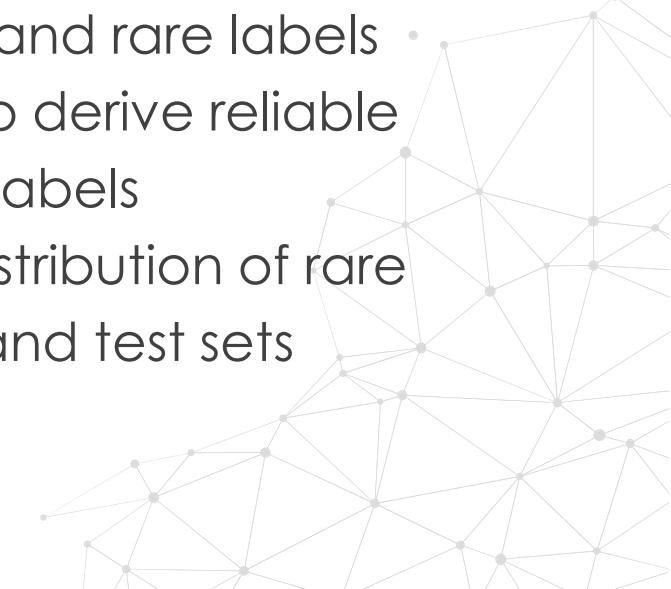
- Strings need to be encoded to numbers for use with Scikit-Learn
- Rare labels may cause over-fitting and operationalisation problems
- Hard to understand the role of the rare label on the outcome prediction
- Removing rare labels may improve model performance



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
 - How to quantify category frequency
 - Examples of frequent and rare labels
 - Example of difficulty to derive reliable information from rare labels
 - Example of uneven distribution of rare labels between train and test sets





THANK YOU

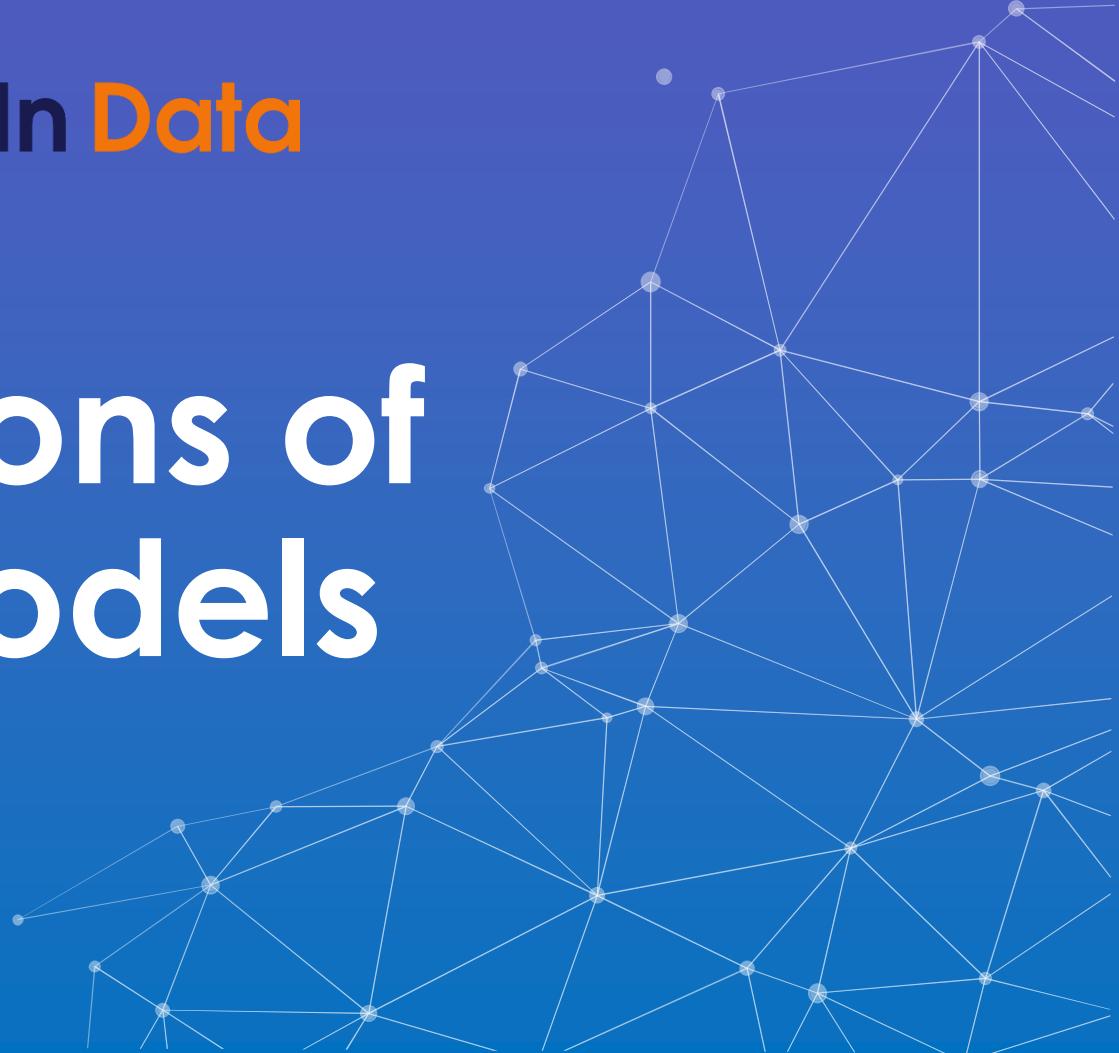
www.trainindata.com





Train In Data

Assumptions of Linear Models



Linear Regression Model

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \dots + \beta_n X_{ni} + \varepsilon_i$$

- Y is the outcome variable
- X are the predictor variables
- β are the coefficients
- β_0 is the intercept
- ε_i is the difference between the predicted and the observed value of Y for the i th observation



Linear Model Assumptions

1. **Linearity:** The mean values of the outcome variable for each increment of the predictor(s) lie along a straight line. There is a linear relationship between predictors and target.
2. **No perfect multicollinearity:** There should be no perfect linear relationship between two or more of the predictors.
3. **Normally distributed errors:** the residuals (ε_i) are random, normally distributed with a mean of 0.
5. **Homoscedasticity:** At each level of the predictor variable(s), the variance of the residual terms should be constant.



When the assumptions are met

- The coefficients and parameters of the regression equation are said to be unbiased.
- The model can be accurately applied.



When the assumptions are not met

The variables are not good enough to predict accurately the outcome.

Some issues could be:

- Outliers
- Lack of homoscedasticity
- The variables are too skewed



• What can we do?

Transforming the data is useful to correct the problems with outliers and homoscedasticity.

- Mathematical transformations
- Discretisation
- Remove or censor outliers

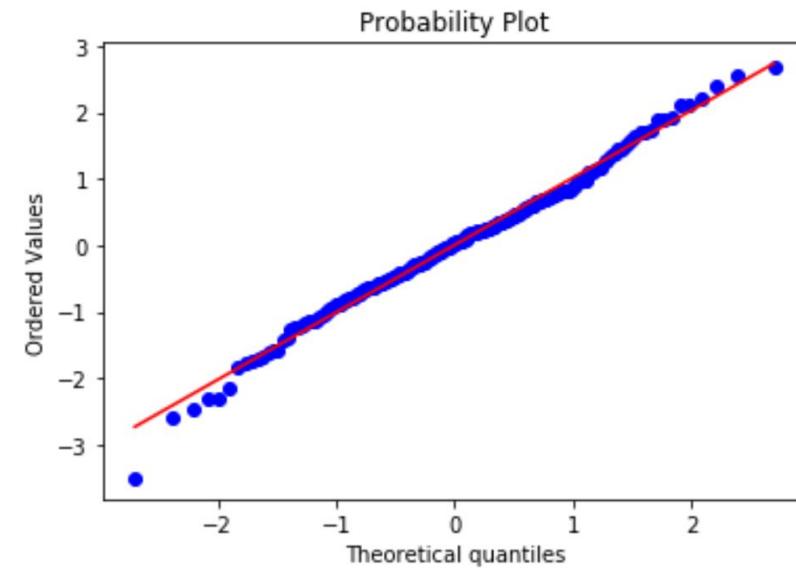
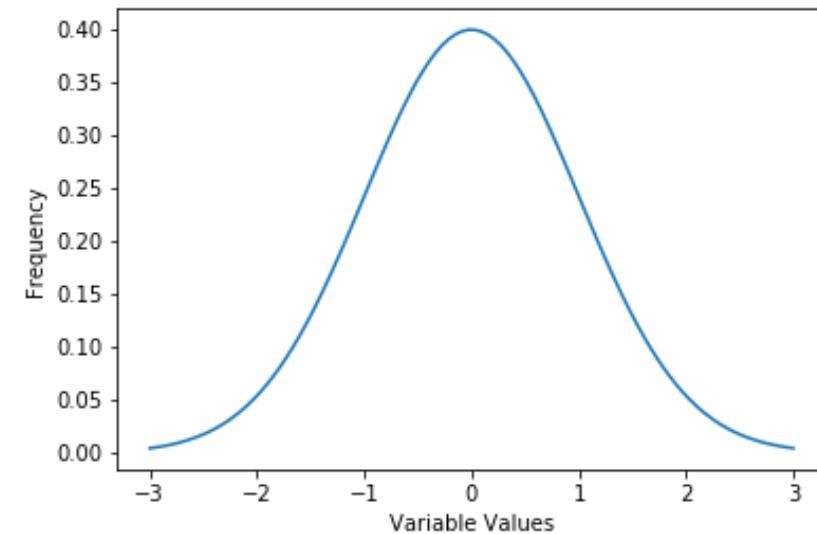


- Evaluate model performance



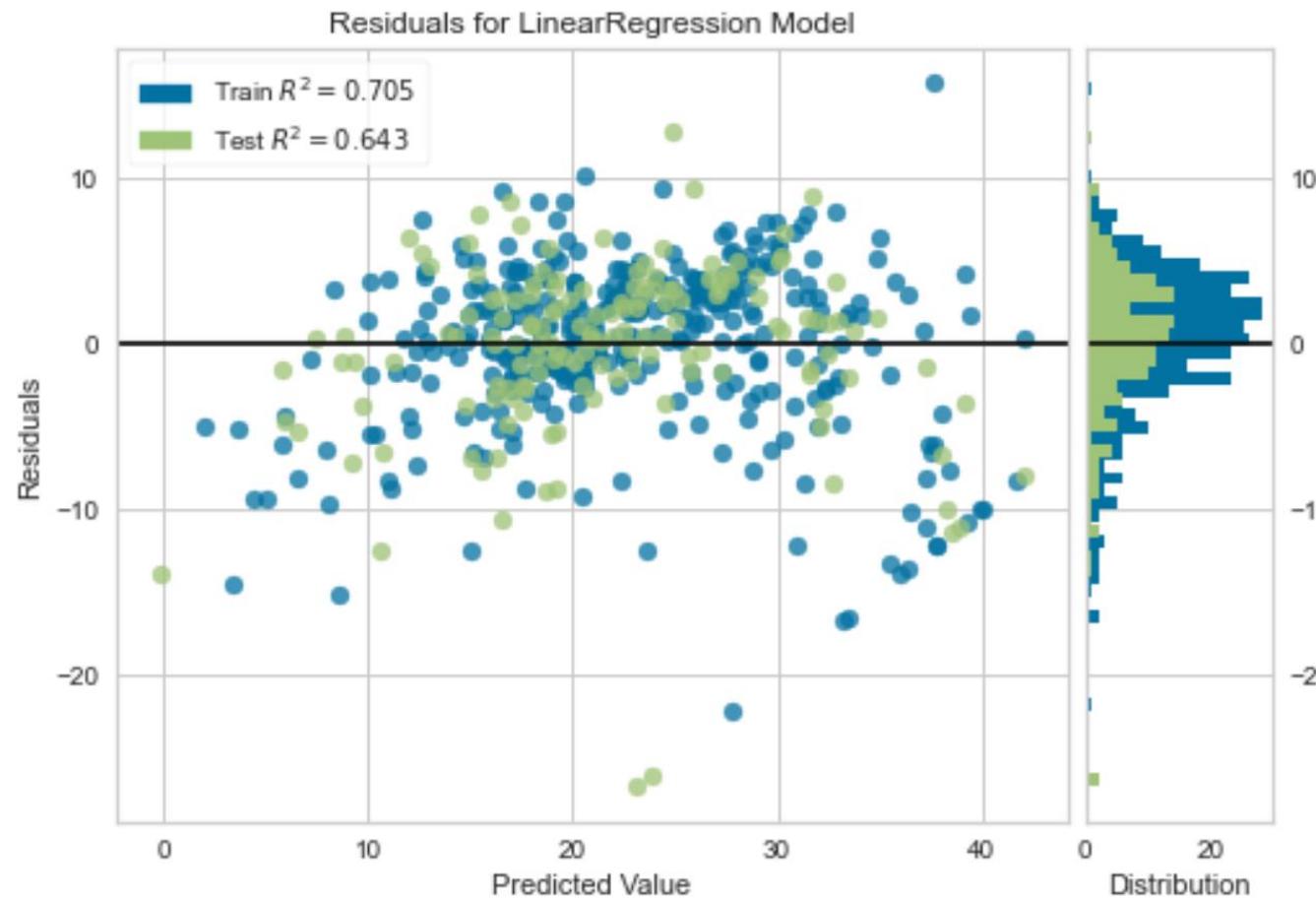
• Errors $\sim N(0, \sigma)$

- Normality can be assessed with histograms and Q-Q plots
- Normality can be statistically tested, for example with the Kolmogorov-Smirnov test.

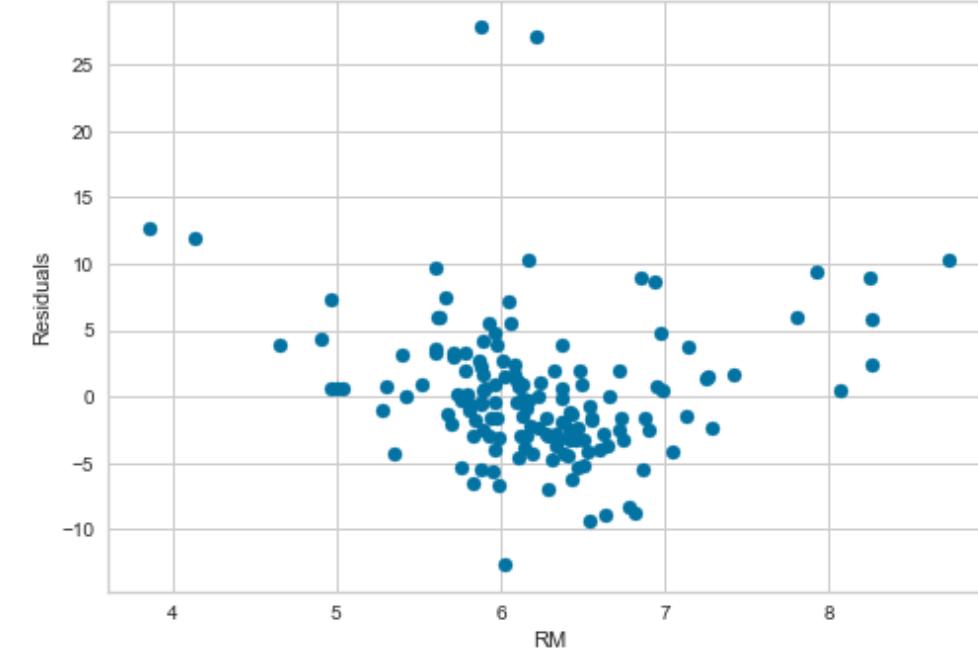
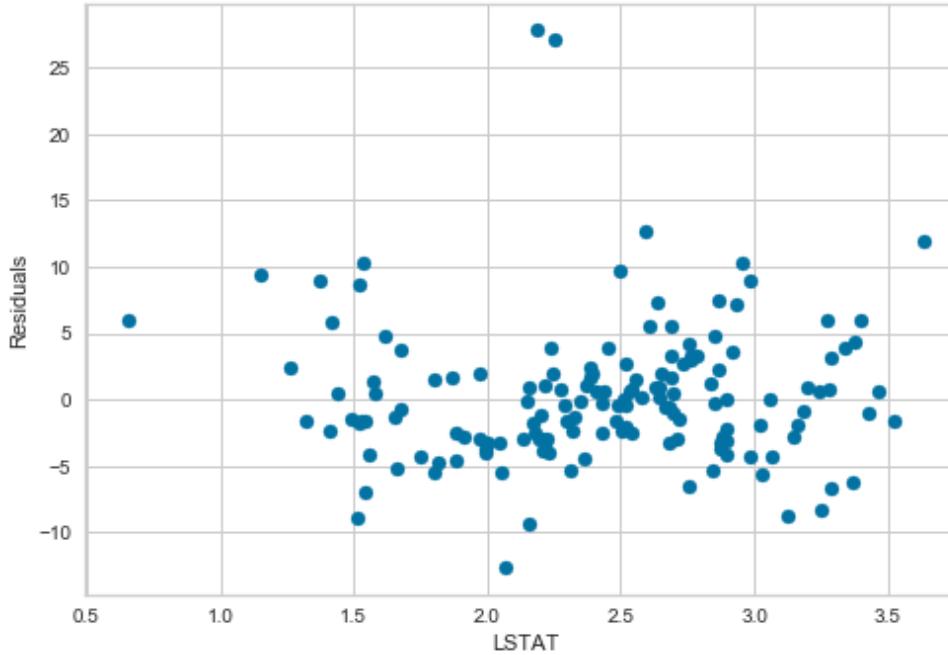


Homoscedasticity

- There are tests and plots to determine homoscedasticity.
 - Residuals plot
 - Levene's test
 - Barlett's test
 - Goldfeld-Quandt Test
- Visual inspection



Homoscedasticity

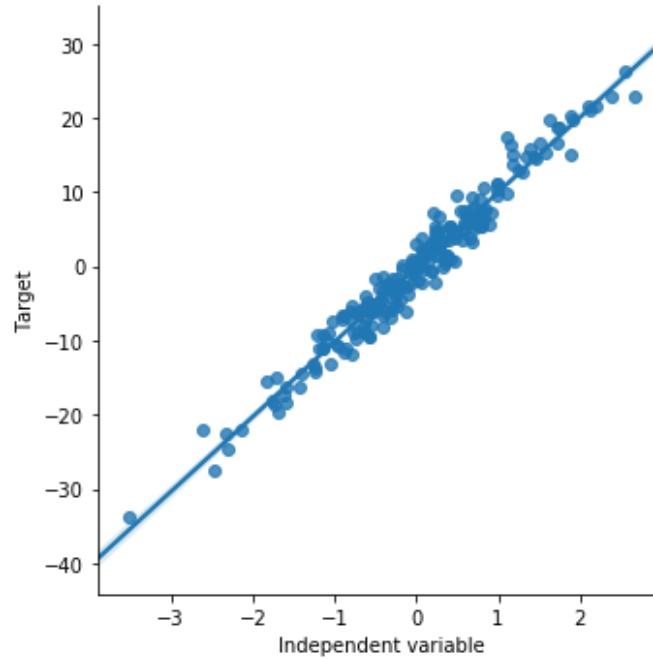


Homoscedasticity: the error term (that is, the “noise” in the relationship between the independent variables X and the dependent variable Y) is the same across all the independent variables.

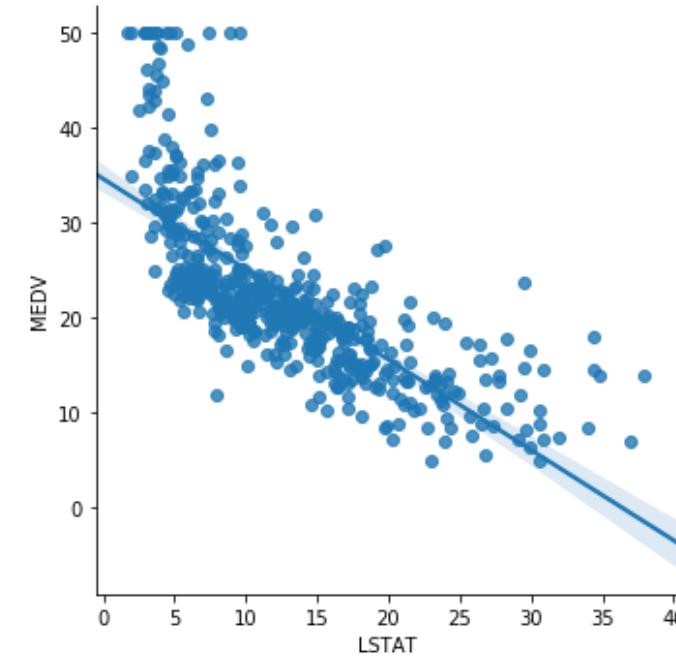
To identify homoscedasticity we need to plot the residuals vs each of the independent variables.

Linear Relationship – Scatter plots

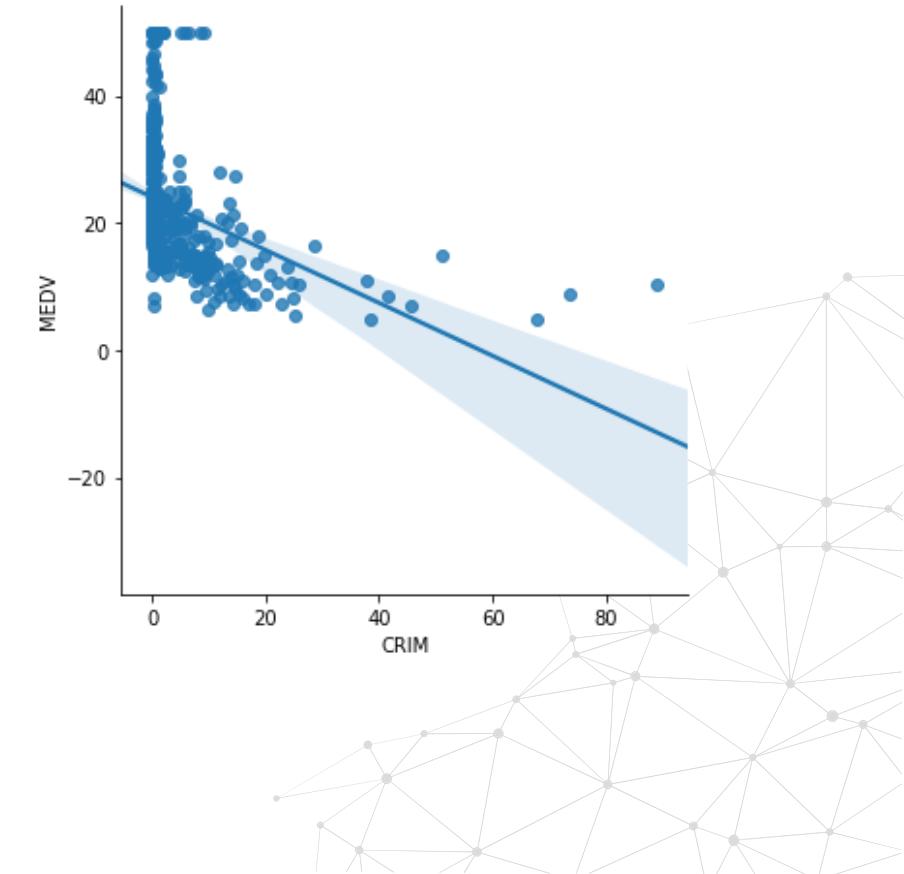
Expected – Simulated data



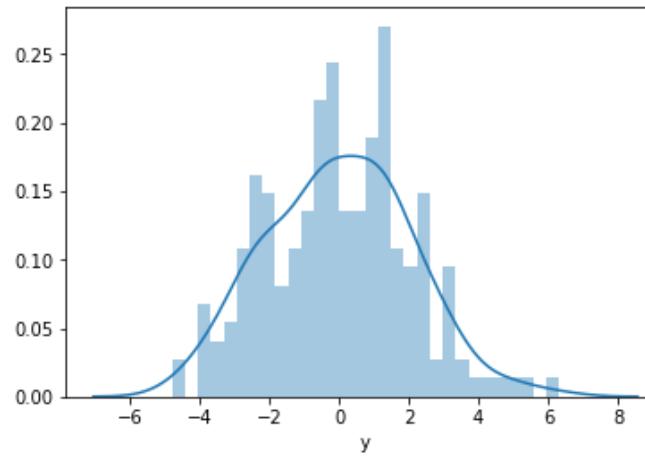
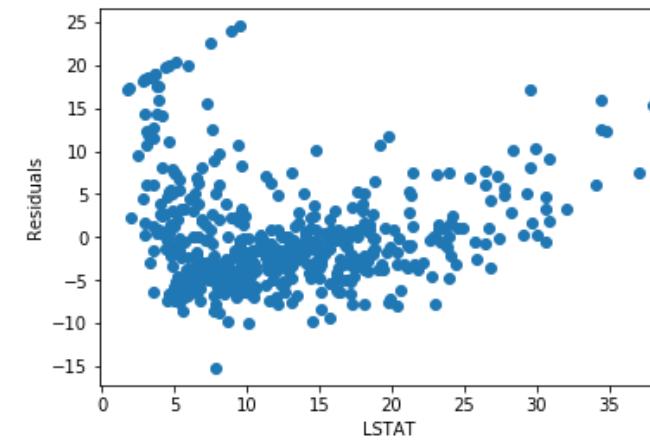
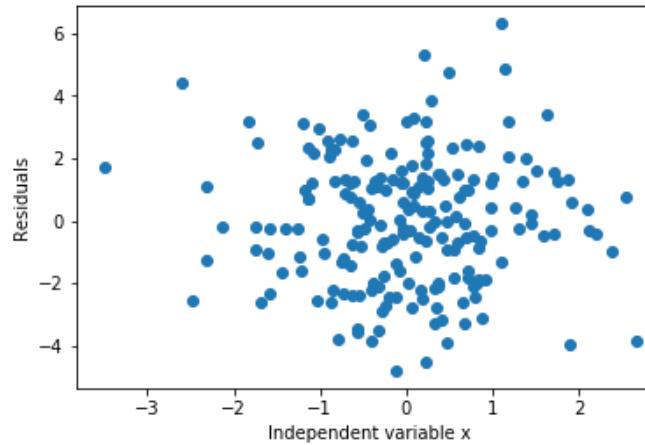
Somewhat linear relationship



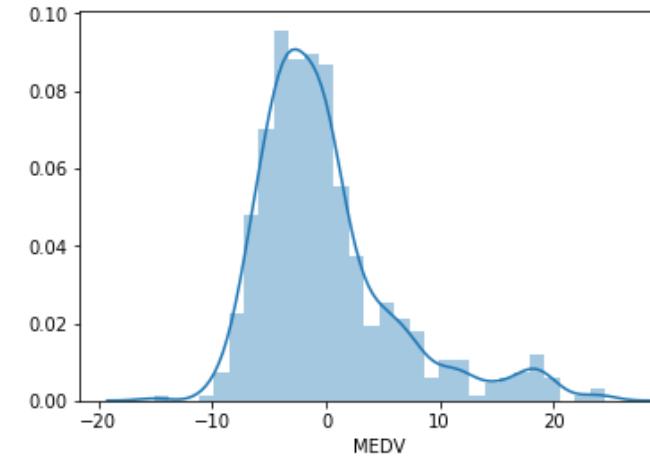
Non-linear relationship



Linear Relationship – Residual plots

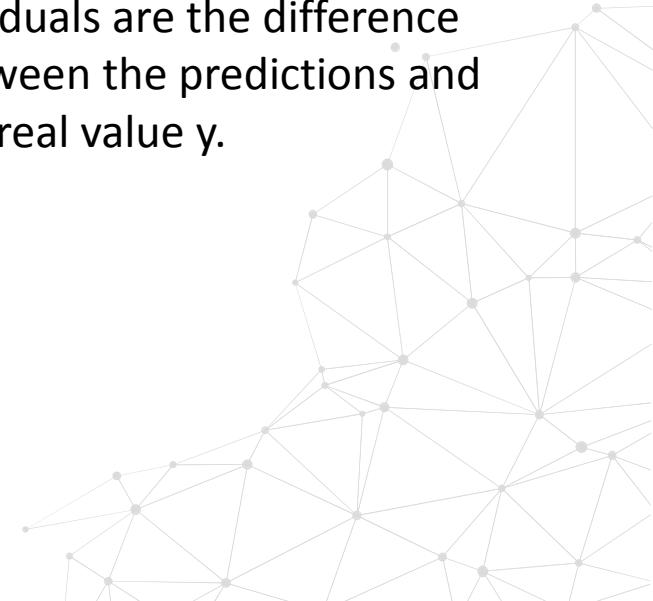


Expected – Simulated data



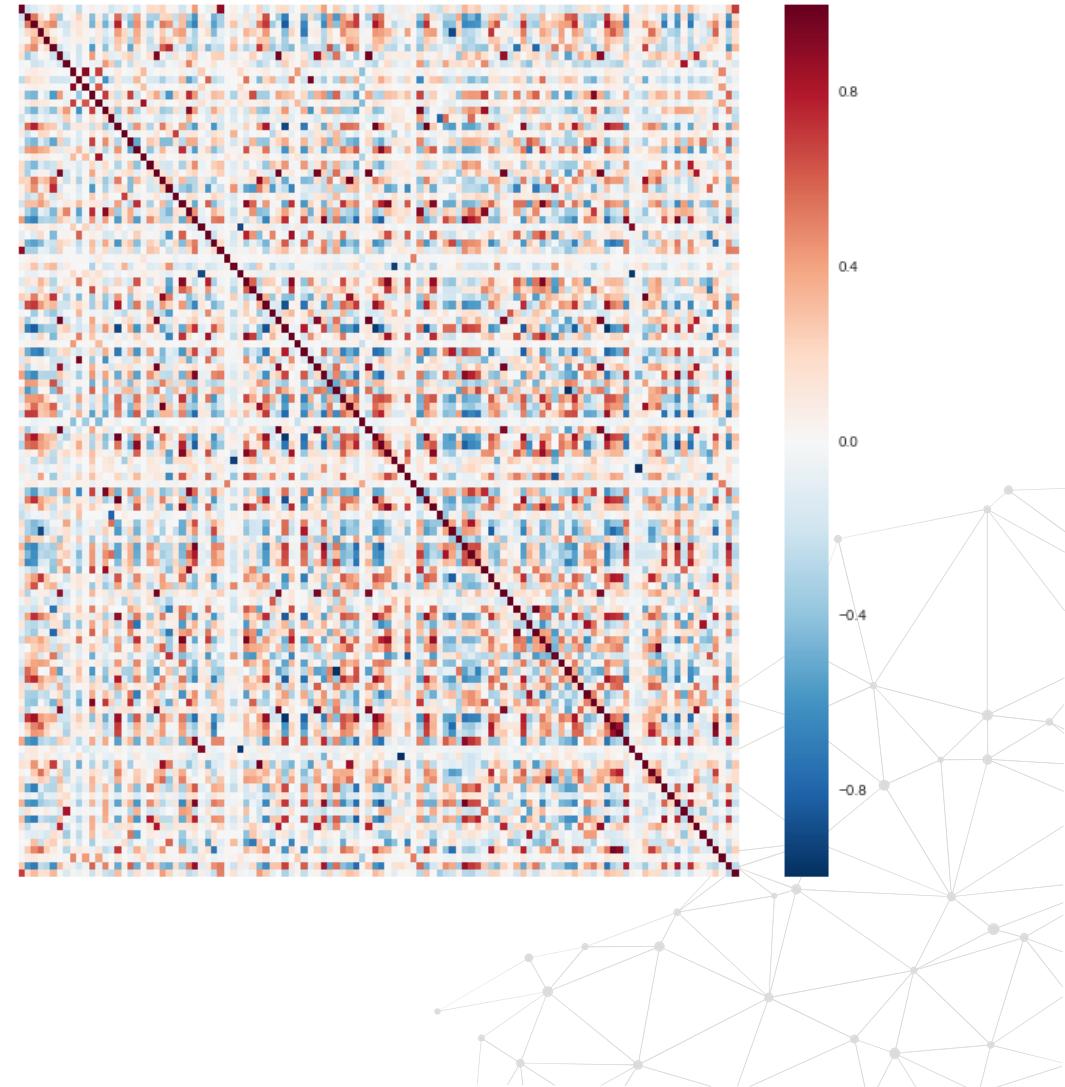
Somewhat linear relationship

- If relationship between X and y is linear, residuals should be normally distributed and centred around 0
- Residuals are the difference between the predictions and the real value y .



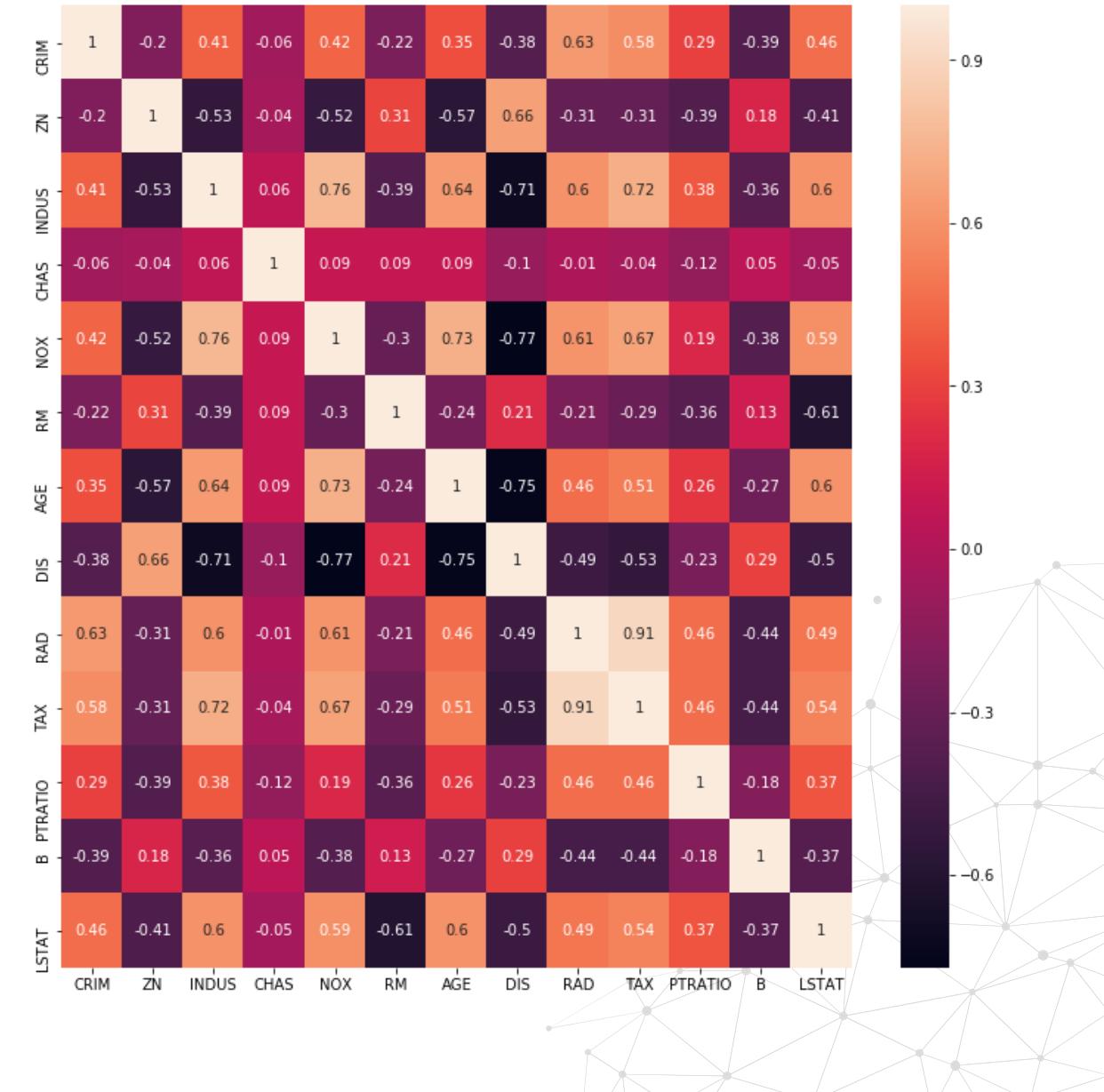
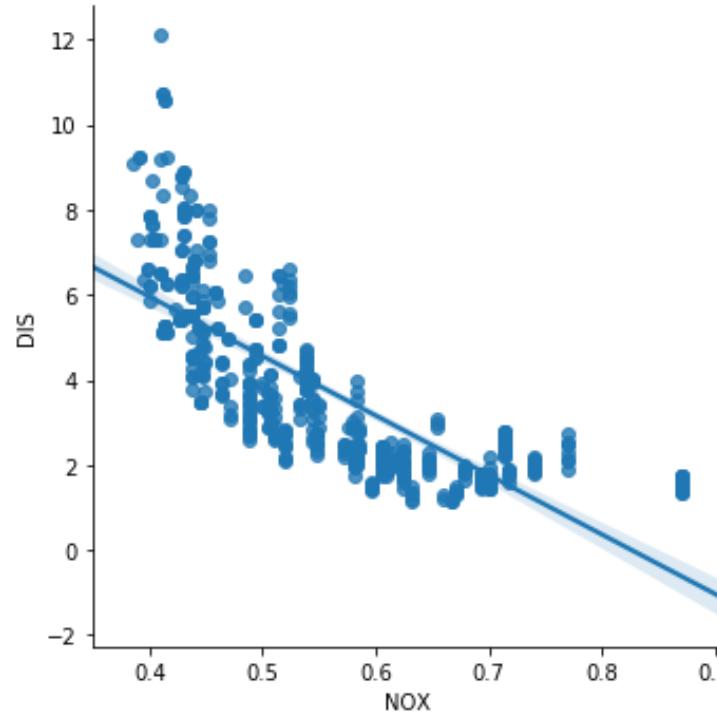
Multicollinearity

- Multicollinearity occurs when the independent variables are correlated with each other
- Multicollinearity can be assessed with a correlation matrix or the variance inflation factor (VIF)
 - Outside of the scope of this course
 - Check the course Feature Selection for Machine Learning



Multi Co-linearity

Evaluated by correlation



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
- Full demonstration of the linear assumptions and the influence of non-linear transformations





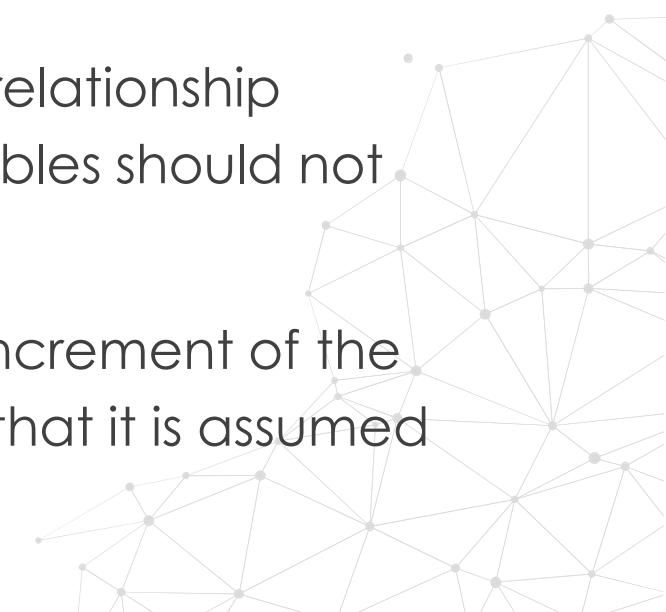
Appendix

More Linear Model assumptions



Linear Model Assumptions

1. **Variable types:** All predictor variables must be quantitative or categorical (with two categories), and the outcome variable must be quantitative, continuous and unbounded.
2. **Non-zero variance:** The predictors should have some variation in value (i.e., they do not have variances of 0).
3. **No perfect multicollinearity:** There should be no perfect linear relationship between two or more of the predictors. So, the predictor variables should not correlate too highly.
4. **Linearity:** The mean values of the outcome variable for each increment of the predictor(s) lie along a straight line. In plain English this means that it is assumed that the relationship we are modelling is a linear one.



Linear Model Assumptions

5. **Normally distributed errors:** It is assumed that the residuals in the model are random, normally distributed variables with a mean of 0.
6. **Homoscedasticity:** At each level of the predictor variable(s), the variance of the residual terms should be constant. Independent errors: For any two observations the residual terms should be uncorrelated (or independent)
7. **Independence:** all of the values of the outcome variable are independent
8. **Independent errors:** For any two observations the residual terms should be uncorrelated (or independent). This is sometimes described as a lack of autocorrelation.





THANK YOU

www.trainindata.com





Probability Distributions

• What is a probability distribution?

- A probability distribution is a function that describes the likelihood of obtaining the possible values that a variable can take.
- For the variable height, the probability distribution describes how often we can get a value of 161 cm, or 174 cm, or 200 cm, etc.
- As you can infer from the previous, it is more likely to obtain values between 161 – 170 cm, than values around or bigger than 200 cm.



Properties of probability distributions

Probability distributions indicate the likelihood of an event or outcome.

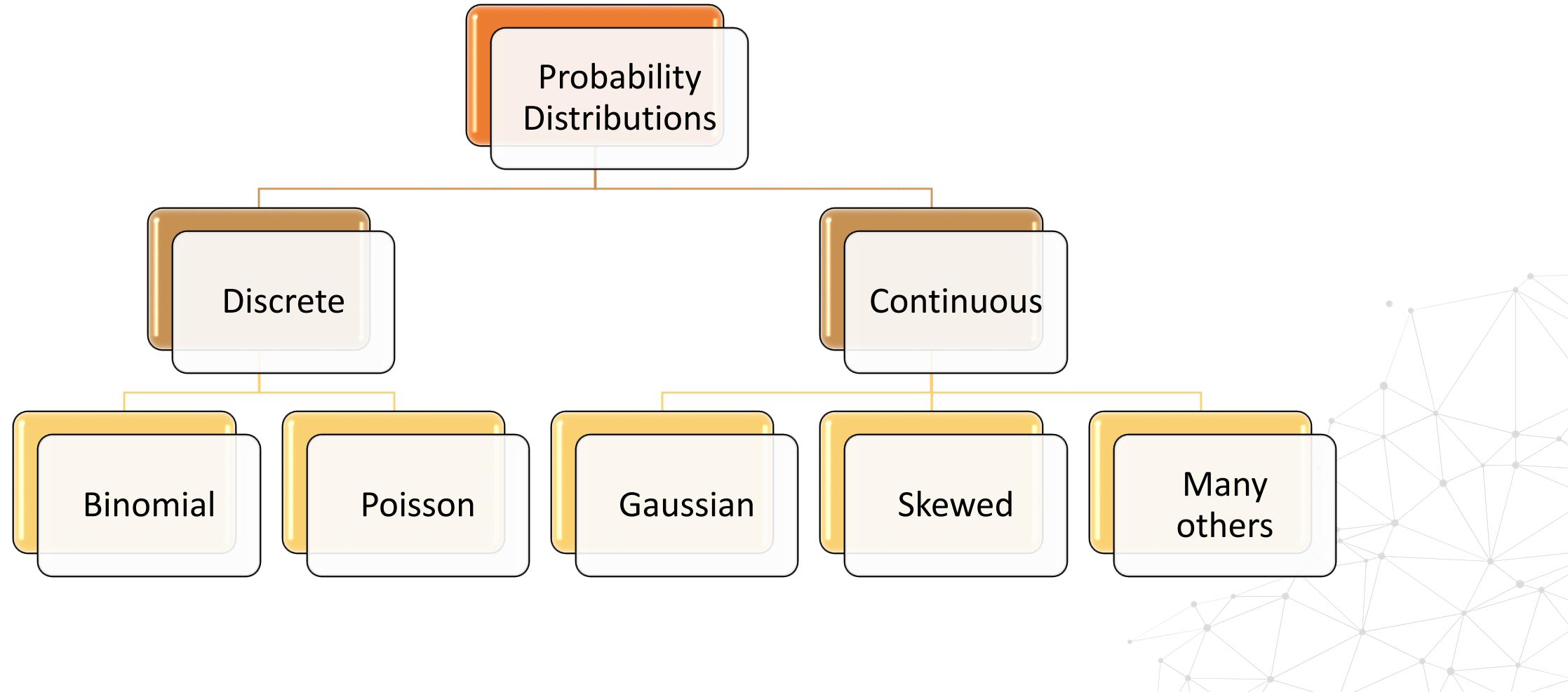
$p(x)$ = the likelihood that random variable takes a specific value of x .

The sum of all probabilities for all possible values must equal 1.

The probability for a particular value or range of values must be between 0 and 1.



Different probability distributions



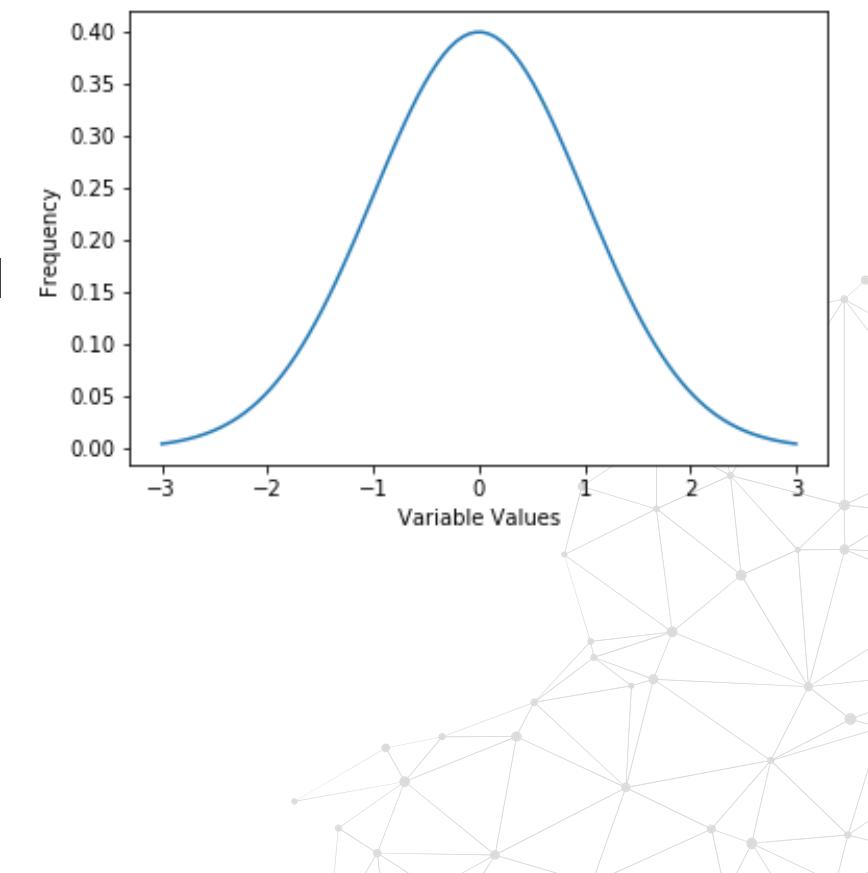
• Gallery of probability distributions

Follow this [link](#) for more probability distributions.



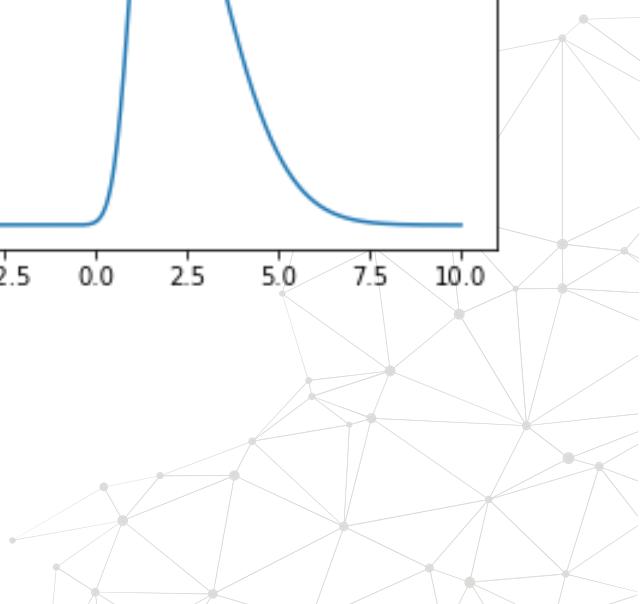
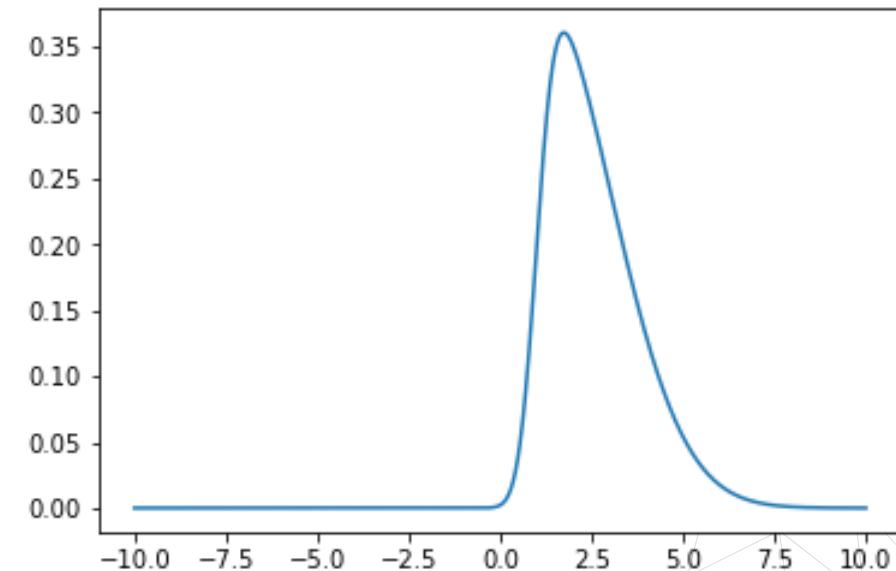
The Normal distribution

- Many natural phenomena follow a normal distribution
 - Height, blood pressure, etc.
- Symmetric:
 - Most of the observations occur around the central peak
 - Probabilities for values further away from the centre decrease equally in both directions.
 - Extreme values in both tails of the distribution are similarly unlikely.

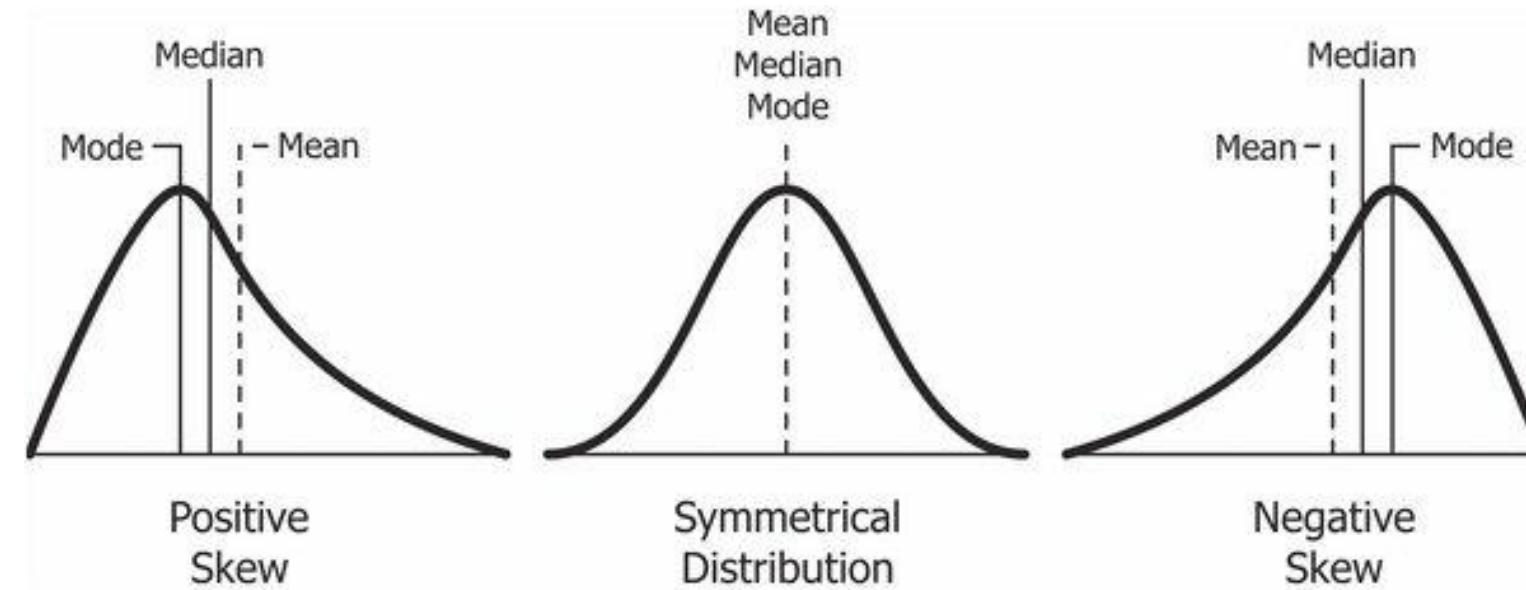


Skewed distributions

- A distribution is skewed if one of its tails is longer than the other
- A left-skewed distribution has a long left tail. Also called negatively-skewed distributions.
- A right-skewed distribution shows a long right tail. Also called positive-skew distributions.



Gaussian vs Skewed distributions

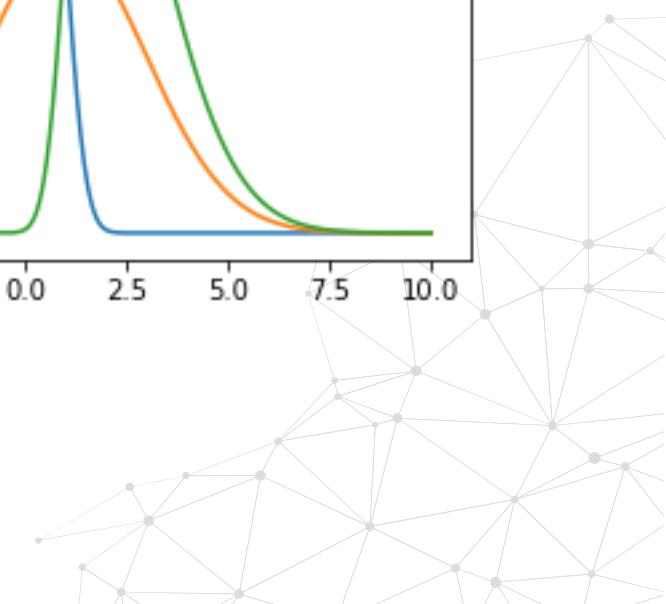
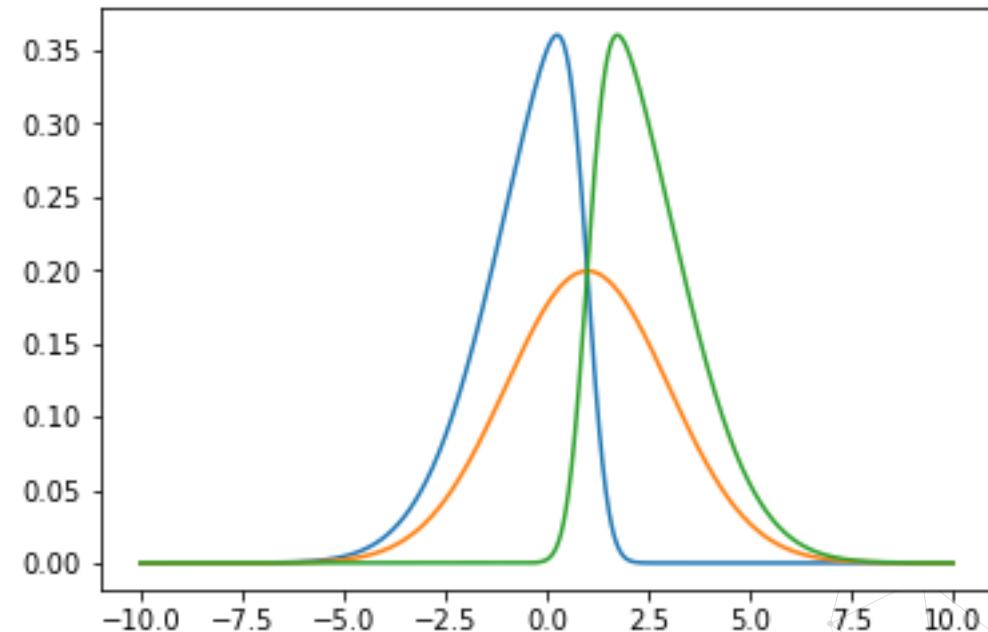


- In Normal distributions, the mean, median and mode are the same
- For skewed distributions, the mean is influenced by the tail

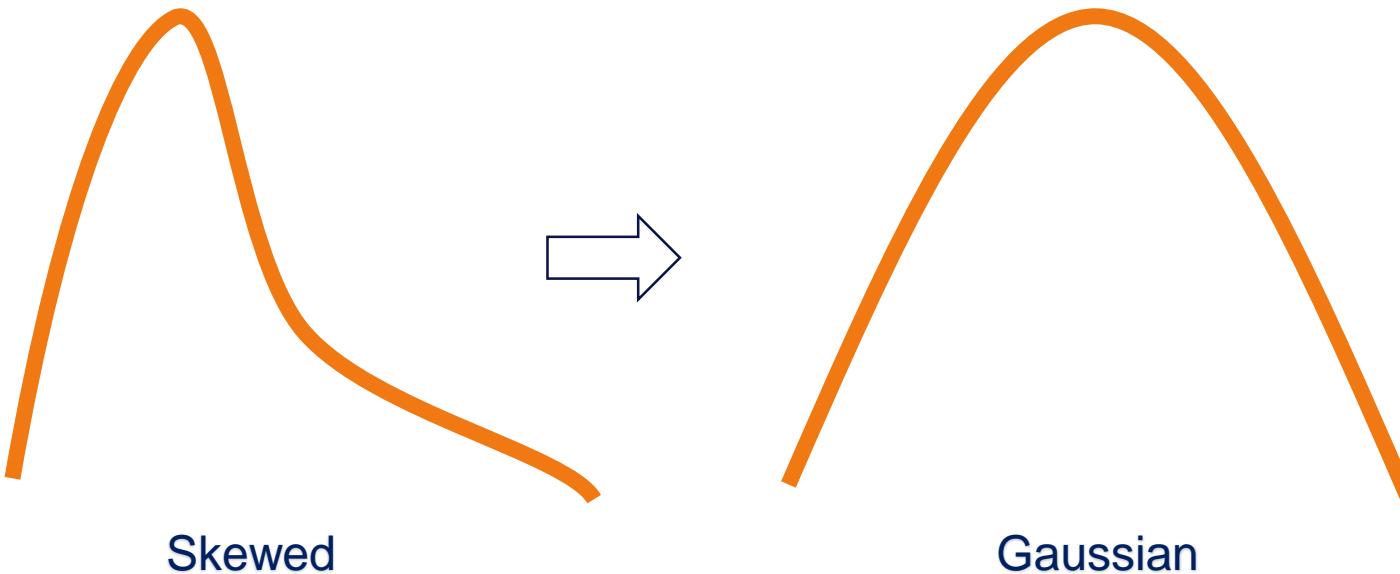


Distributions and model performance

- Linear Models assume that the residuals are normally distributed.
- Other models make no assumption in the distribution of the variables, however a better spread of the values may improve their performance



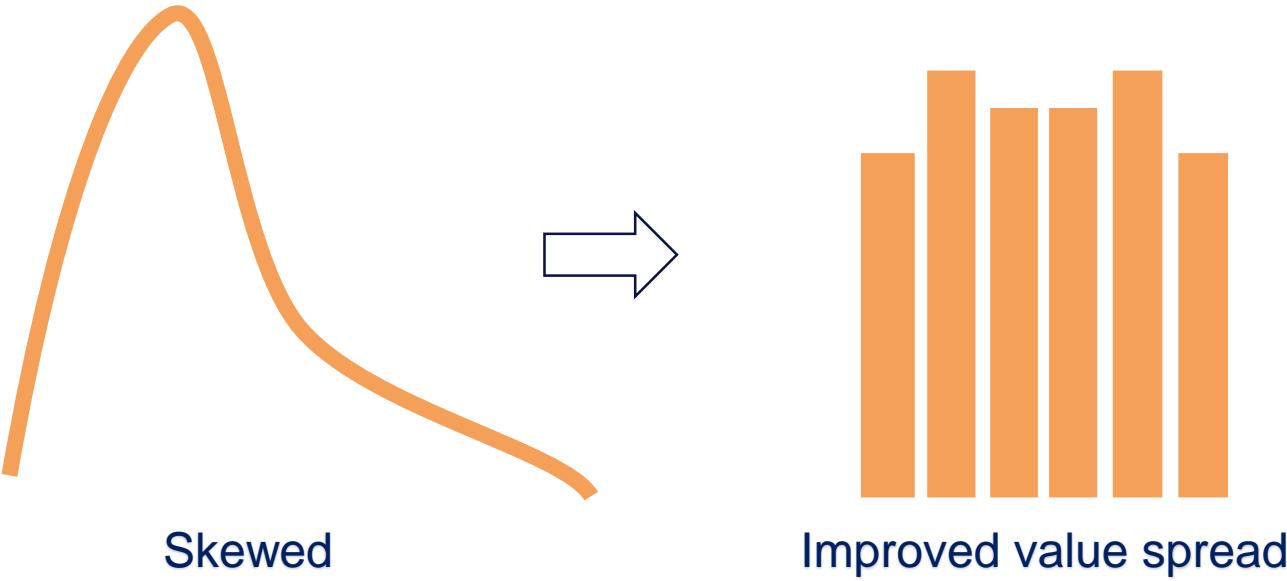
• Variable transformation



Variable transformation

- Logarithmic $\rightarrow \ln(x)$
- Exponential $\rightarrow x^{\text{Exp}} (\text{any power})$
- Reciprocal $\rightarrow (1 / x)$
- Box-Cox $\rightarrow (x^{\text{Exp}}(\lambda) - 1) / \lambda$
 - λ varies from -5 to 5

• Variable discretisation



Discretisation

- Equal width bins
 - Bins $\rightarrow (\max - \min) / n$ bins
 - Generally does not improve the spread
- Equal frequency bins
 - Bins determined by quantiles
 - Equal number of observations per bin
 - Generally improves spread





THANK YOU

www.trainindata.com





Train In Data

Outliers

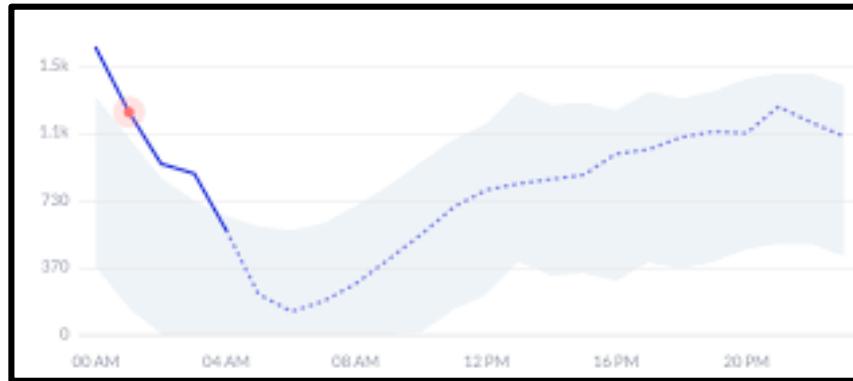
Outliers

- An outlier is a data point which is significantly different from the remaining data.
- “An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.” [D. Hawkins. Identification of Outliers, Chapman and Hall , 1980.]

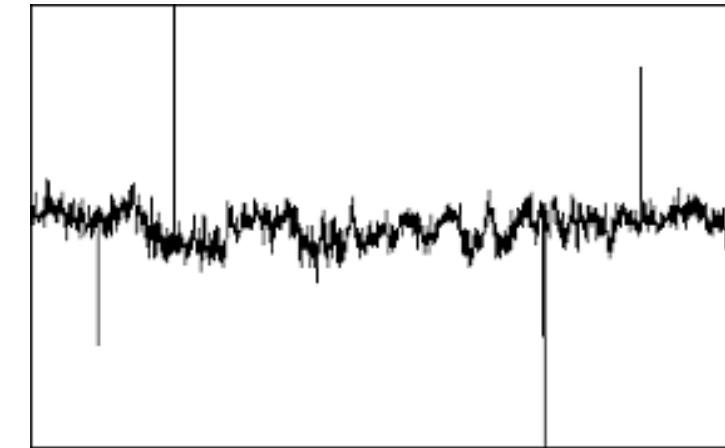


• Should outliers be removed?

Revenue forecasting



Credit card transactions



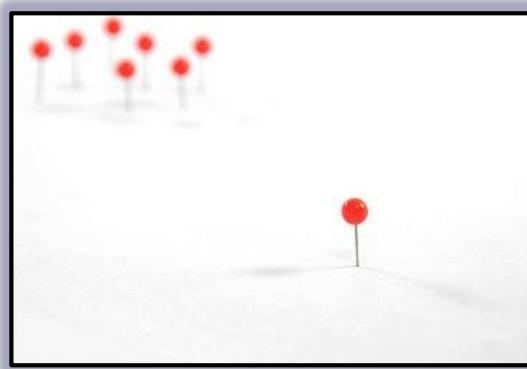
Depending on the context, outliers either deserve special attention or should be completely ignored.

Approach to outliers in this course

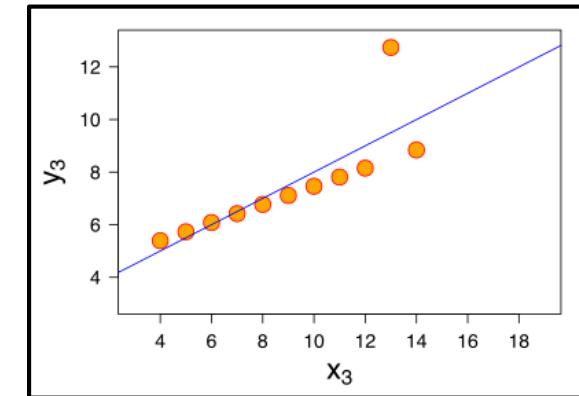
- Handle outliers in cases where they may affect model performance
- The course is tailored to improve model performance
- Out of scope: outlier detection
 - A massive field with lots of techniques



Algorithms susceptible to outliers



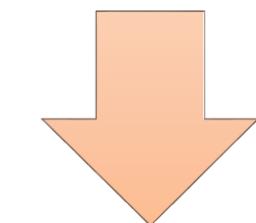
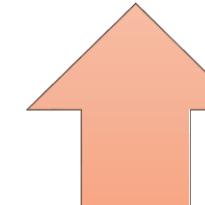
Linear
models



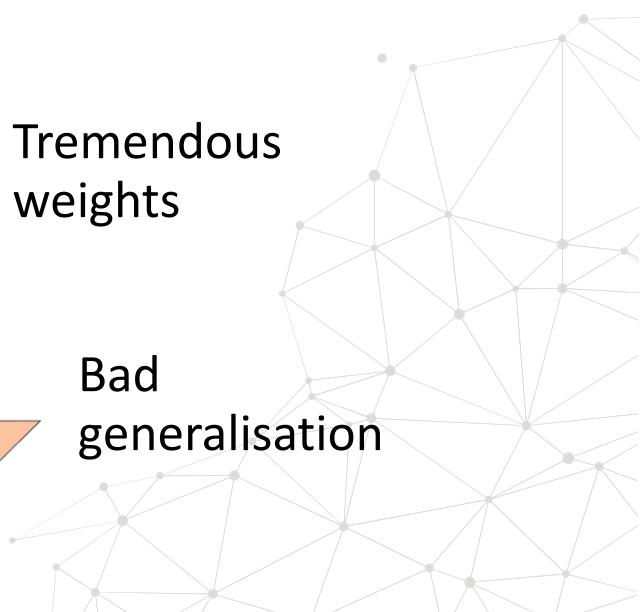
Adaboost



Tremendous
weights



Bad
generalisation



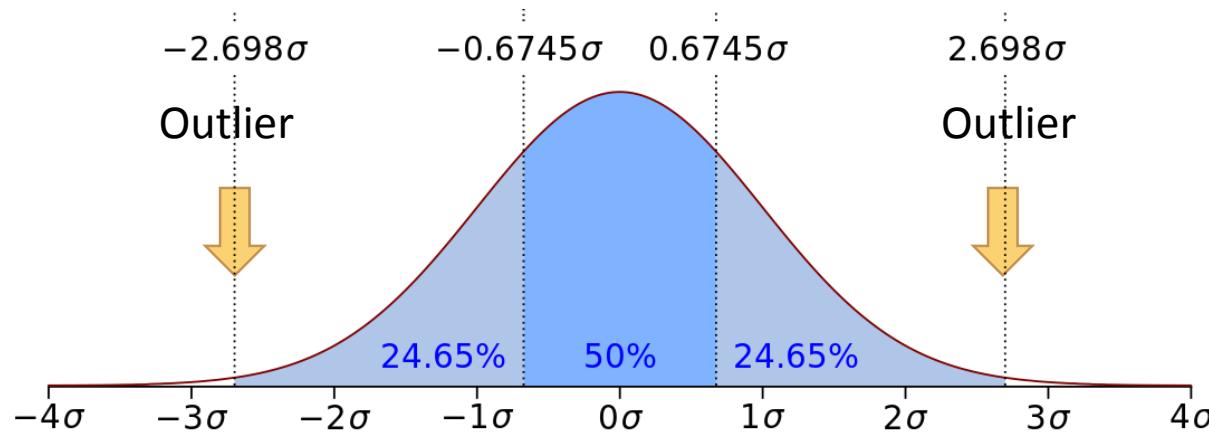


Detecting Outliers

Extreme Value Analysis



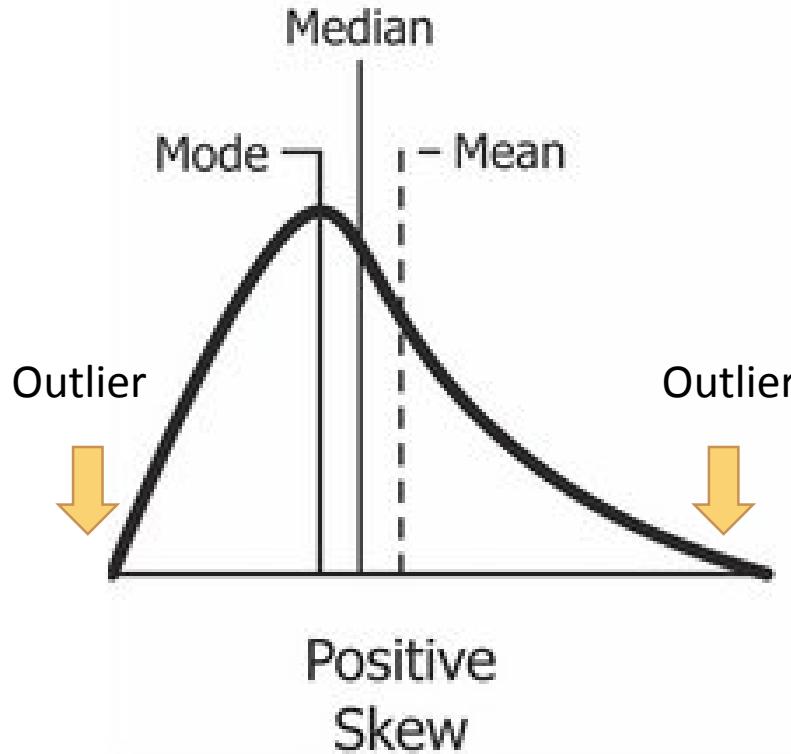
Normal distribution



- ~99% of the observations of a normally distributed variable lie within the mean $\pm 3 \times$ standard deviations.
- Values outside mean $\pm 3 \times$ standard deviations are considered outliers

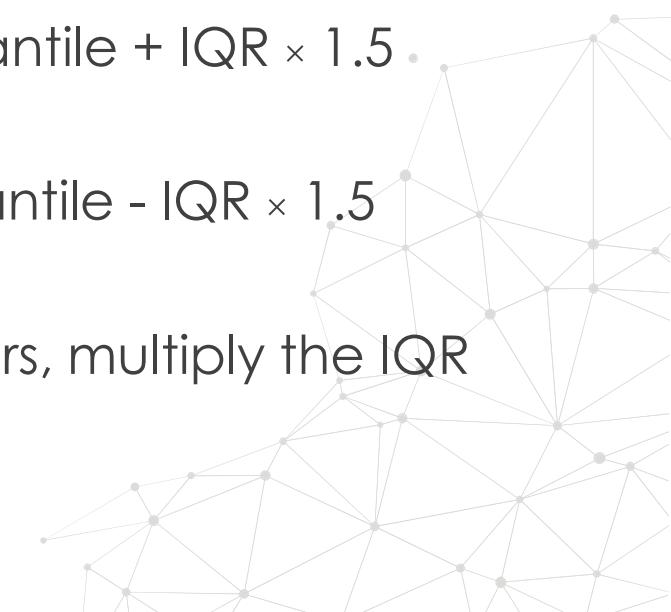


Skewed distributions



- The general approach is to calculate the quantiles, and then the inter-quantile range (IQR), as follows:
- $IQR = 75^{\text{th}} \text{ Quantile} - 25^{\text{th}} \text{ Quantile}$
- $\text{Upper limit} = 75^{\text{th}} \text{ Quantile} + IQR \times 1.5$.
- $\text{Lower limit} = 25^{\text{th}} \text{ Quantile} - IQR \times 1.5$

Note, for extreme outliers, multiply the IQR by 3 instead of 1.5

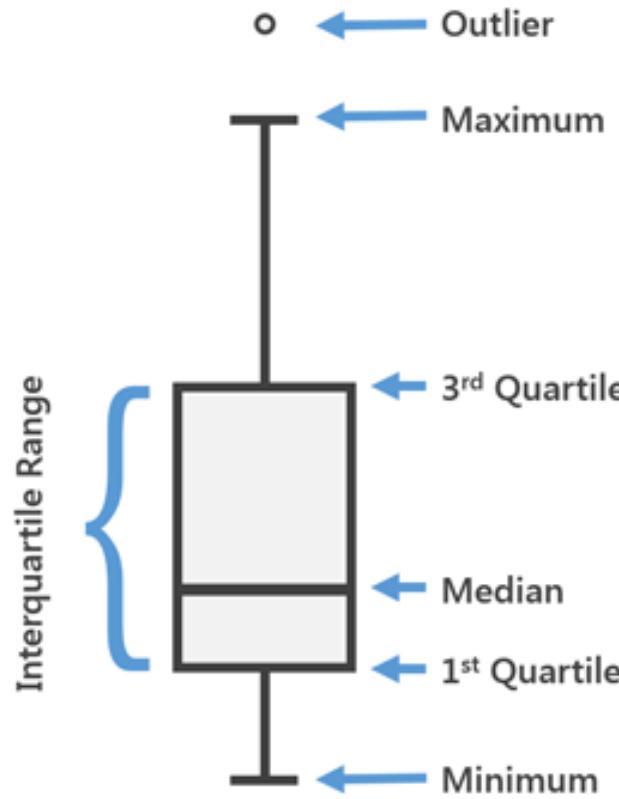


Notes on quantiles

- Quartiles = dividing the distribution in 4
- Quantiles = dividing the distribution into 100
- 1st Quartile = 25th Quantile
- 3rd Quartile = 75th Quantile
- 2nd Quartile = 50th Quantile = Median
- IQR = 75th Quantile – 25th Quantile = 3rd Quartile – 1st Quartile

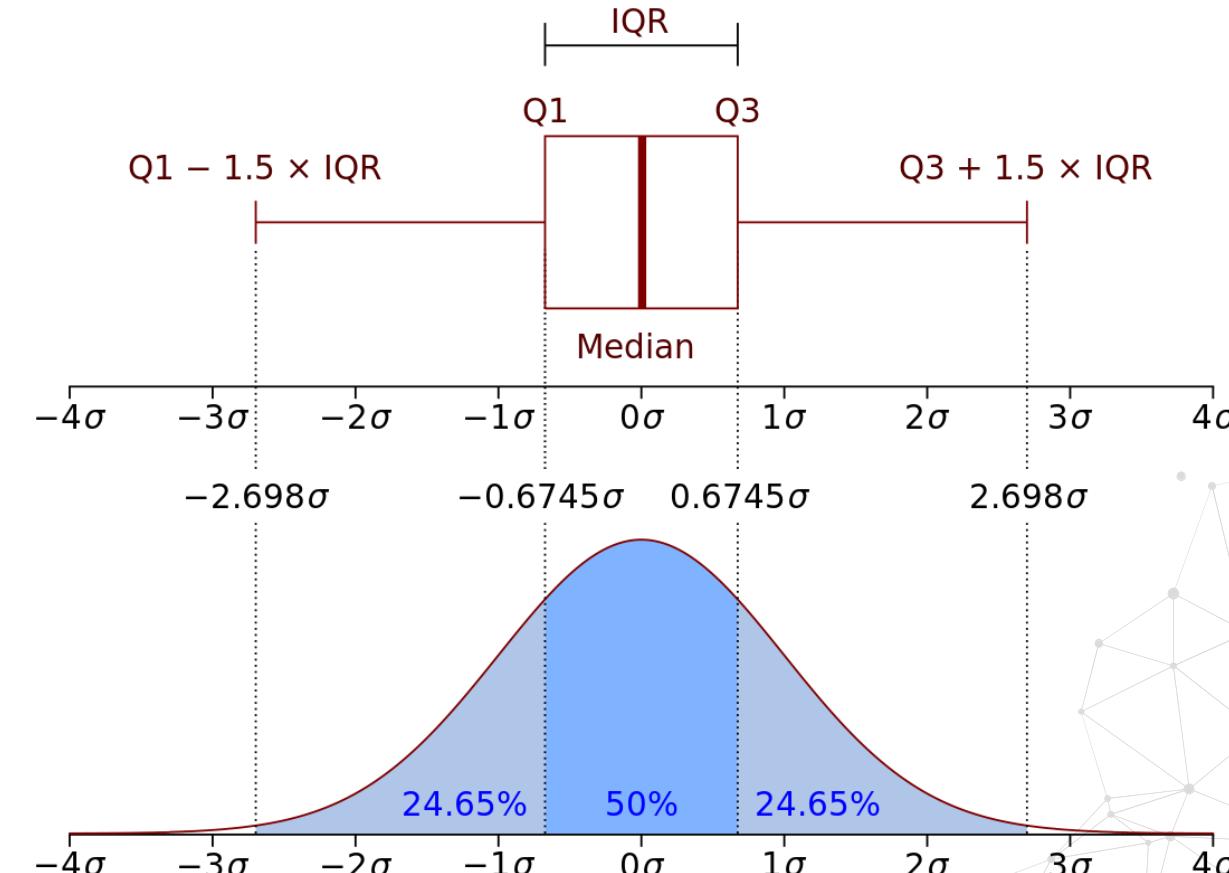
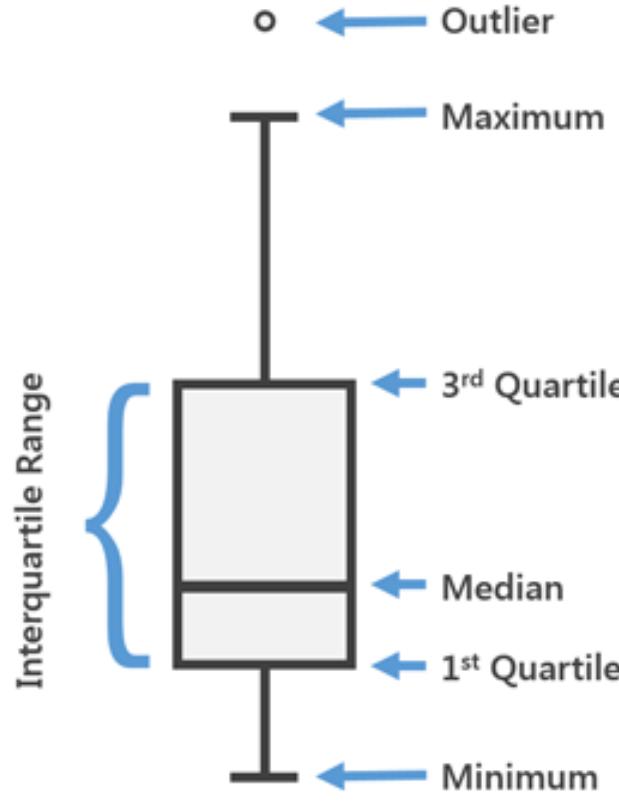


Visualising outliers - Boxplots



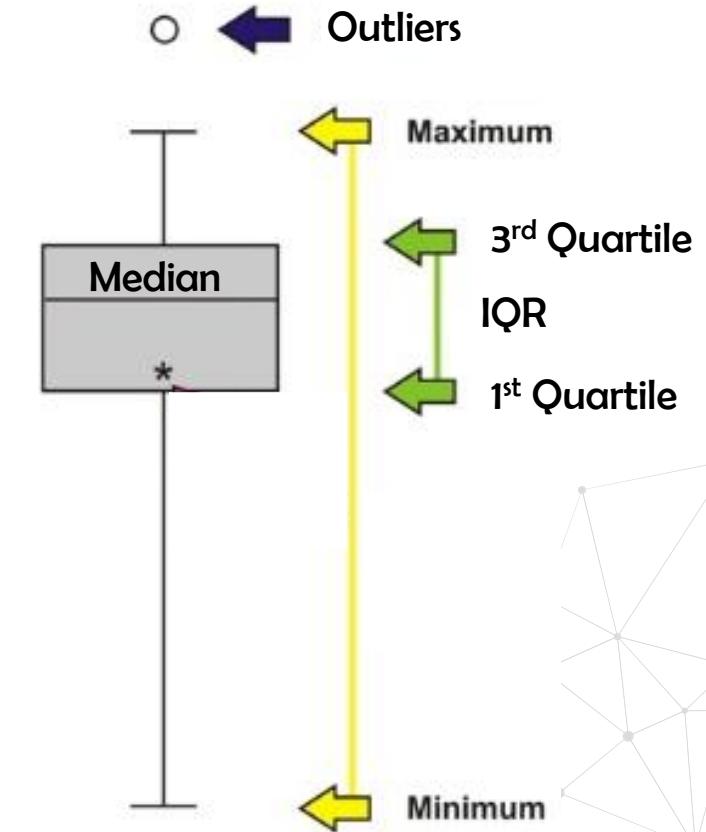
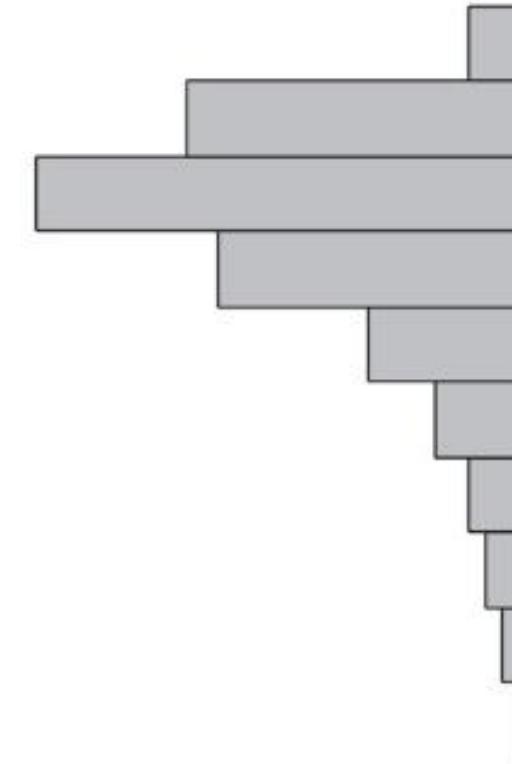
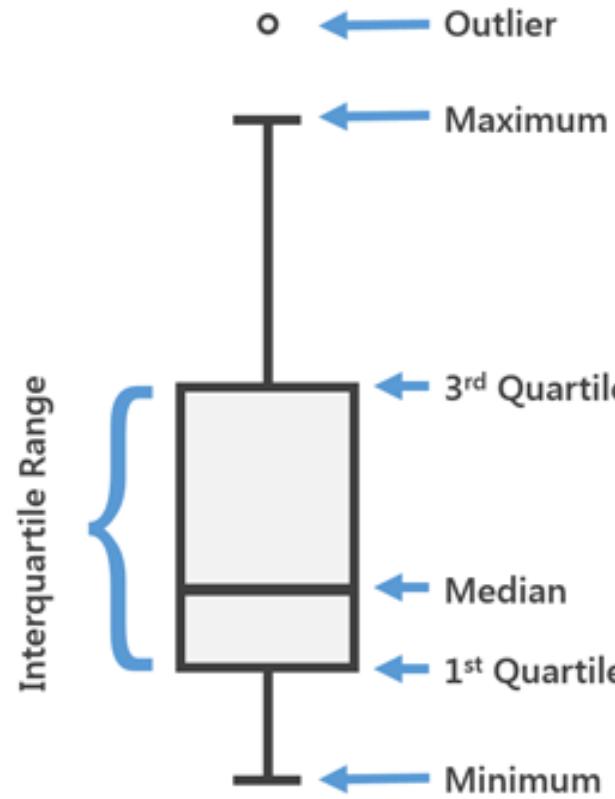
Images taken from pro.arcgis.com and wiki.commons

Visualising outliers - Boxplots



Images taken from pro.arcgis.com and [wiki.commons](https://commons.wikimedia.org)

Visualising outliers - Boxplots

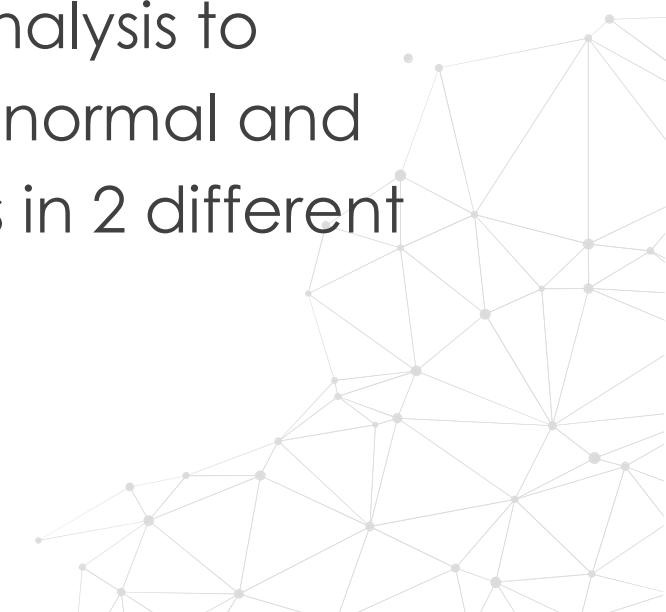


Images taken from pro.arcgis.com and wiki.commons

• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
- Extreme Value Analysis to detect outliers in normal and skewed variables in 2 different datasets





THANK YOU

www.trainindata.com



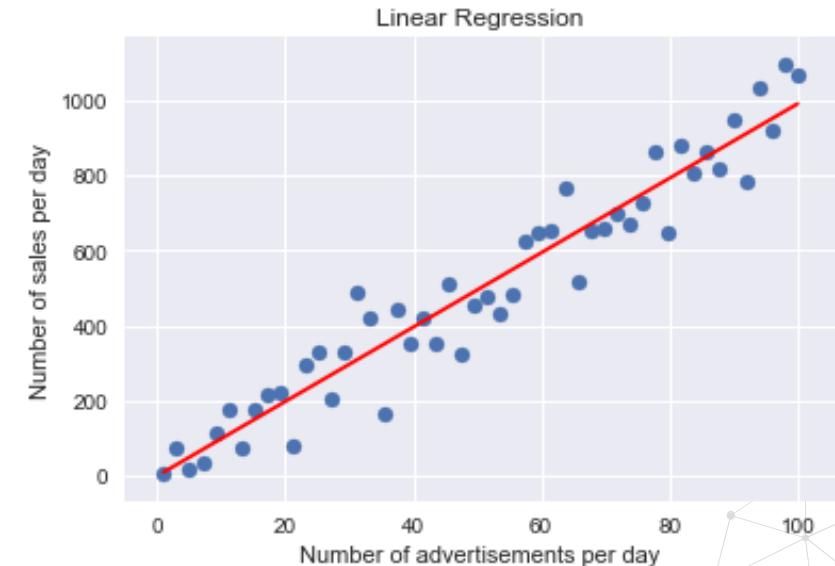


Feature Magnitude

Linear Models

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n$$

- β indicates the change in Y per unit change of X
- If X changes scale, β will change its value
- Regression coefficients depend of the magnitude of the variable
- Features with bigger magnitudes dominate over features with smaller magnitudes

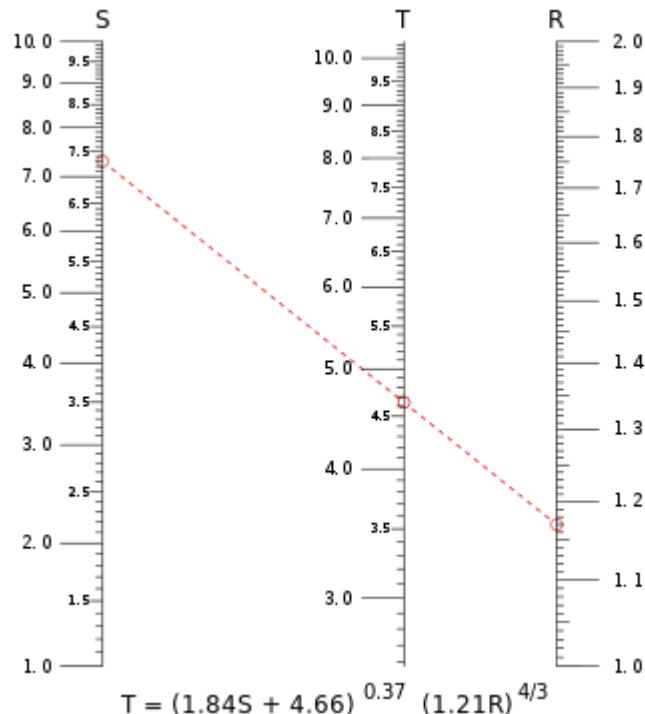


• Feature Magnitude matters

- The regression coefficient is directly influenced by the scale of the variable
- Variables with bigger magnitude / value range dominate over the ones with smaller magnitude / value range
- Gradient descent converges faster when features are on similar scales
- Feature scaling helps decrease the time to find support vectors for SVMs
- Euclidean distances are sensitive to feature magnitude.



Algorithms sensitive to magnitude

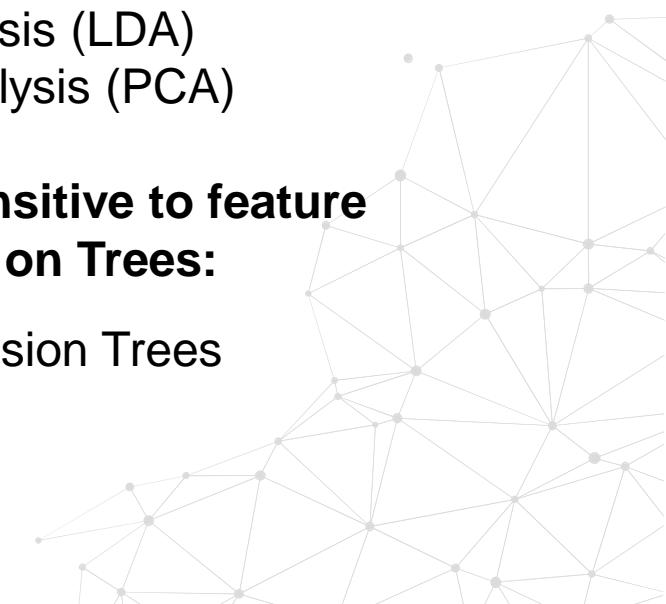


The machine learning models affected by the magnitude of the feature:

- Linear and Logistic Regression
- Neural Networks
- Support Vector Machines
- KNN
- K-means clustering
- Linear Discriminant Analysis (LDA)
- Principal Component Analysis (PCA)

Machine learning models insensitive to feature magnitude are the ones based on Trees:

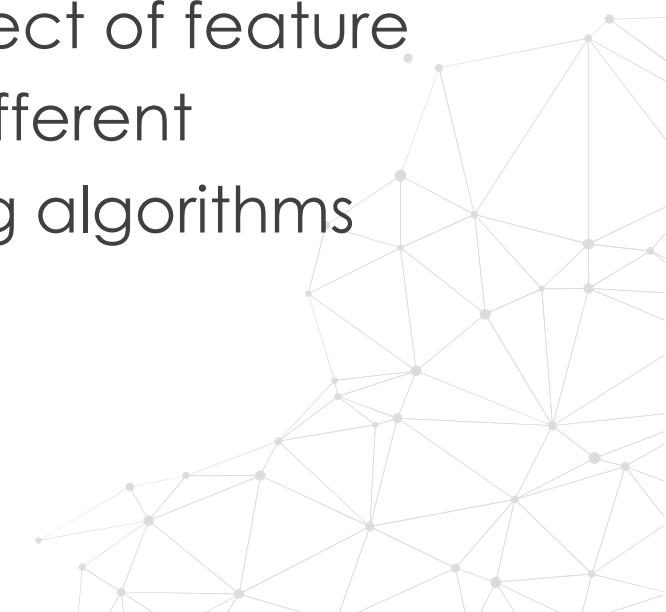
- Classification and Regression Trees
- Random Forests
- Gradient Boosted Trees



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
- Demo on the effect of feature magnitude on different machine learning algorithms





THANK YOU

www.trainindata.com





Missing Data Imputation

Missing Data Imputation

- Imputation is the act of replacing missing data with statistical estimates of the missing values.
- The goal of any imputation technique is to produce a **complete dataset** that can be used to train machine learning models.



Missing Data Imputation Techniques

Numerical Variables



- Mean / Median Imputation
- Arbitrary value imputation
- End of tail imputation

Categorical Variables



- Frequent category imputation
- Adding a “missing” category

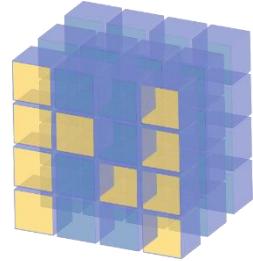
Both



- Complete Case Analysis
- Adding a “Missing” indicator
- Random sample imputation



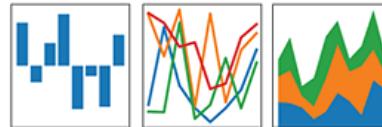
Missing Data Imputation Techniques



NumPy

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Feature-engine

Objectives

Understand the different techniques for missing data imputation.

- Learn multiple techniques
- Understand their impact on the variable and the machine learning model
- Learn how to implement it with pandas, Scikit-learn, and Feature-engine, within a machine learning pipeline



Section Structure

Three main sections:

- Learn multiple techniques (pandas and NumPy)
- Implement the technique with Scikit-learn
- Implement the technique with Feature-engine



Content



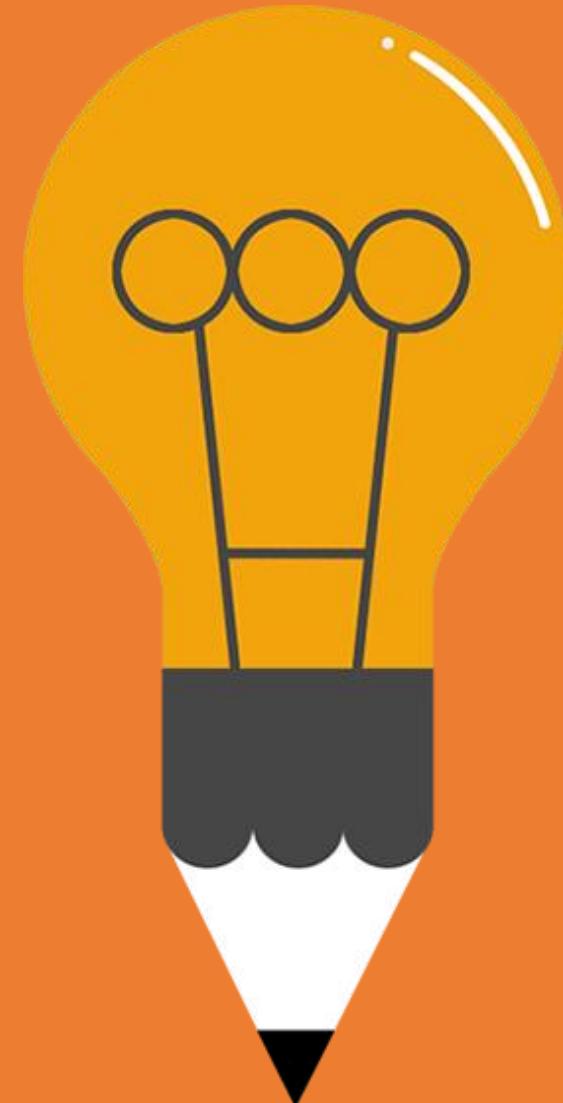
For each lecture:

- Presentation and video
- Accompanying Jupyter notebook
 - Examples of the variable characteristics in real datasets
 - Code to identify and the different variable characteristics



Final Summary

- Final article summarizing how the different variable characteristics affect the different machine learning models at the end of the section.
- Additional reading resources.





THANK YOU

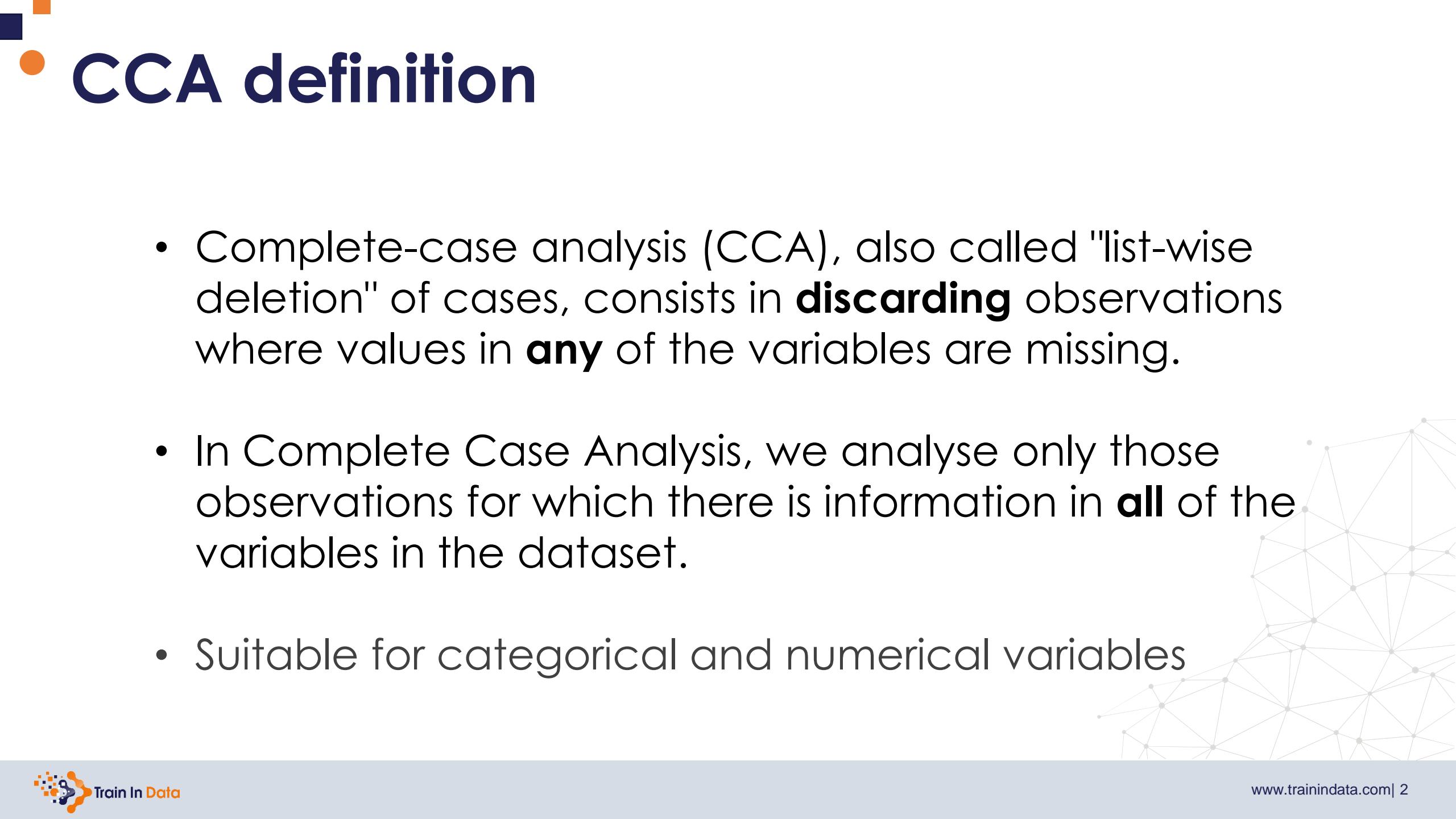
www.trainindata.com





Train In Data

Complete Case Analysis



• CCA definition

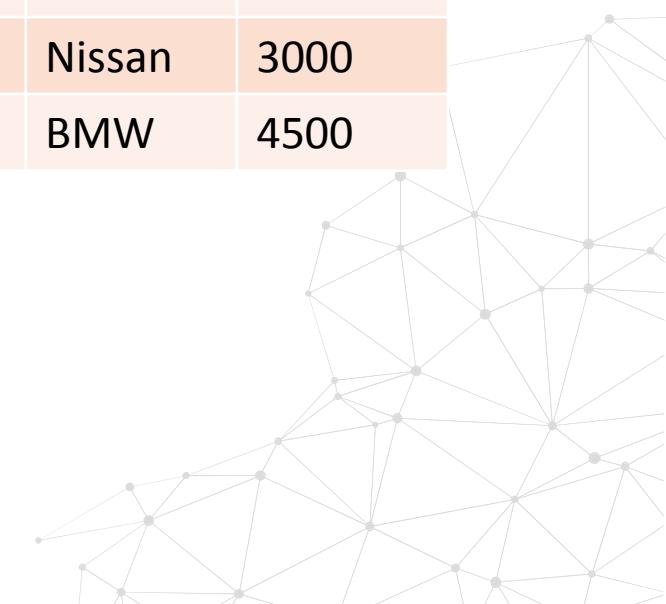
- Complete-case analysis (CCA), also called "list-wise deletion" of cases, consists in **discarding** observations where values in **any** of the variables are missing.
- In Complete Case Analysis, we analyse only those observations for which there is information in **all** of the variables in the dataset.
- Suitable for categorical and numerical variables

• CCA definition

Gender	Price	Make	Engine
Female	100	Ford	2000
	90	Ford	2000
Male	50	Kia	1500
Male	60	Kia	
Female	120	Nissan	3000
Female		BMW	4500
Male	200	BMW	4500



Gender	Price	Make	Engine
Female	100	Ford	2000
Male	50	Kia	1500
Female	120	Nissan	3000
Male	200	BMW	4500



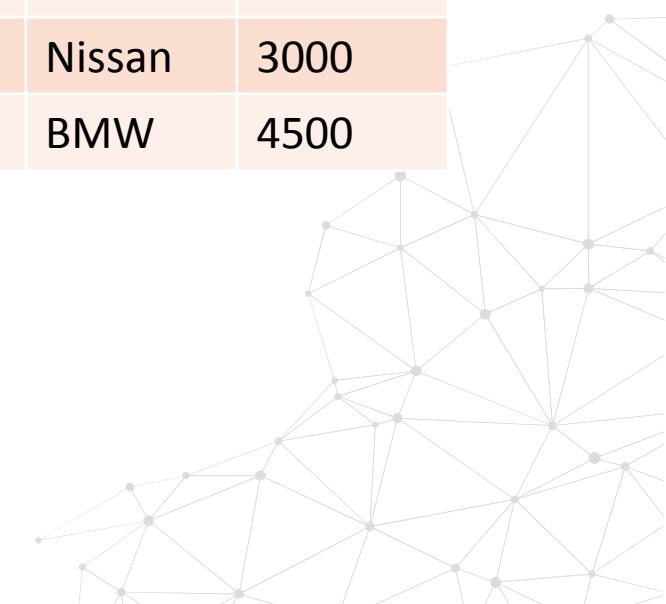
• CCA definition

Gender	Price	Make	Engine
Female	100	Ford	2000
	90	Ford	2000
Male	50	Kia	1500
Male	60	Kia	
Female	120	Nissan	3000
		BMW	4500
Male	200	BMW	4500



Gender	Price	Make	Engine
Female	100	Ford	2000
Male	50	Kia	1500
Female	120	Nissan	3000
Male	200	BMW	4500

- Observations with missing values are removed



CCA Assumptions

- Data is missing at random



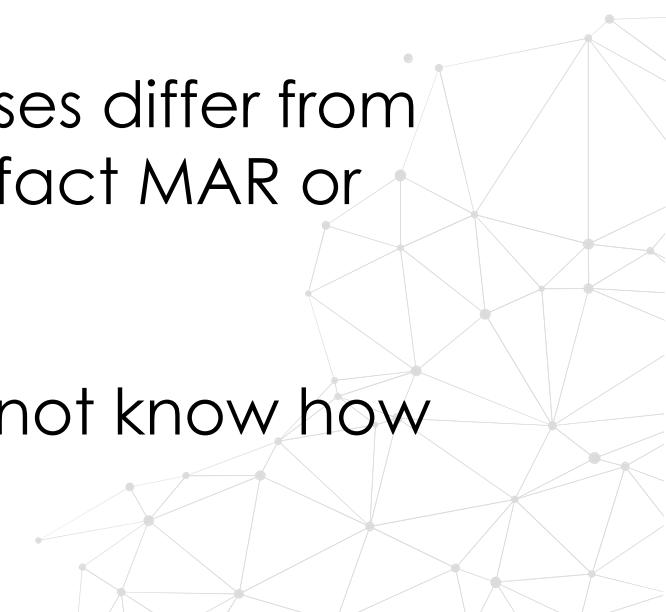
• CCA Advantages

- Simple
- No data manipulation required
- Preserves the distribution of the variables



• CCA Limitations

- It can exclude a large fraction of the original dataset (if missing data is abundant)
- Excluded observations could be informative for the analysis (if data is not missing at random)
- CCA will create a biased dataset if the complete cases differ from the original data (e.g., when missing information is in fact MAR or NMAR and not missing at random).
- When using our models in production, the model will not know how to handle missing data



When to use CCA

- Data is missing completely at random
- No more than 5% of the total dataset contains missing data



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook





THANK YOU

www.trainindata.com





Mean or Median Imputation

• Mean / Median imputation: definition

- Mean / median imputation consists of replacing all occurrences of missing values (NA) within a variable by the mean or median
- Suitable numerical variables



• Mean / Median imputation: example

Price
100
90
50
40
20
100
60
120
200

Mean = 86.66

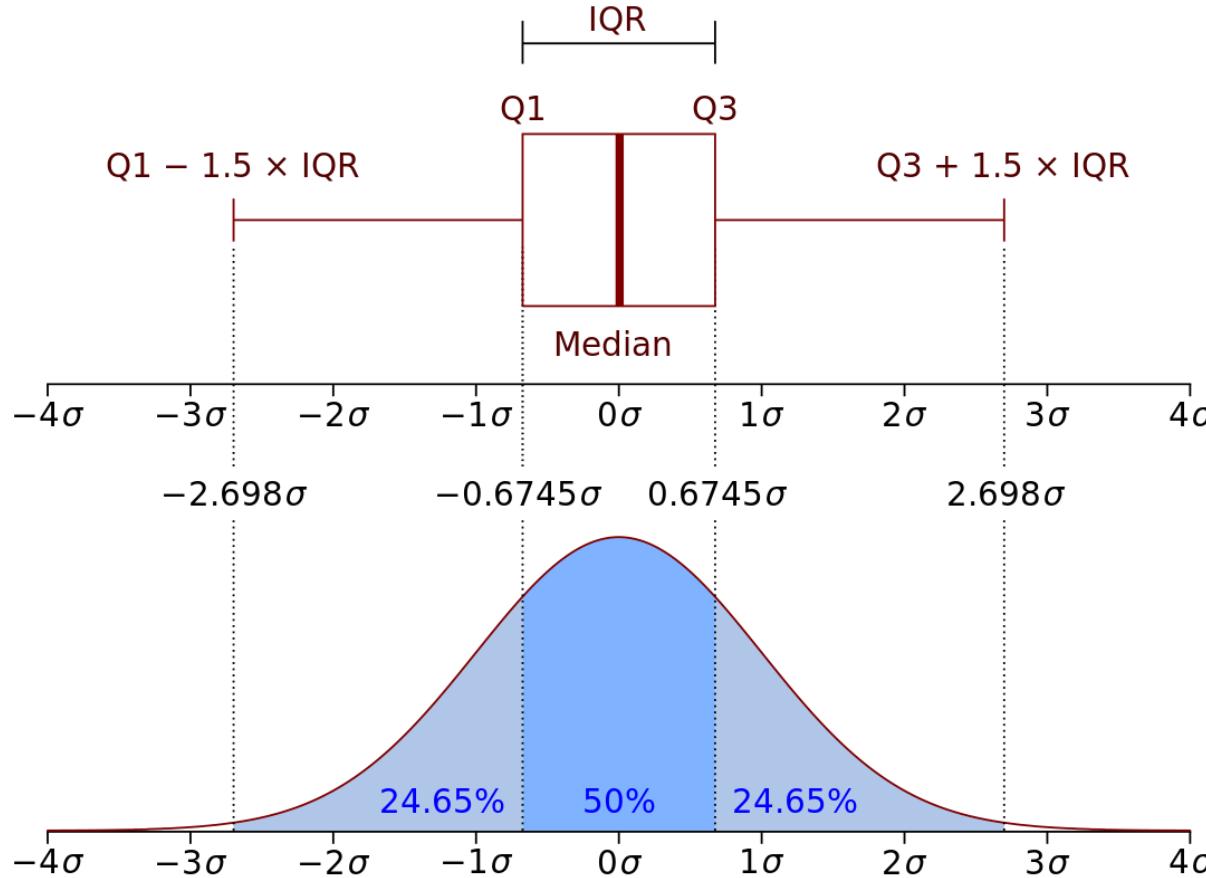
Median = 90



Price
100
90
50
40
20
100
86.66
60
120
86.66
200



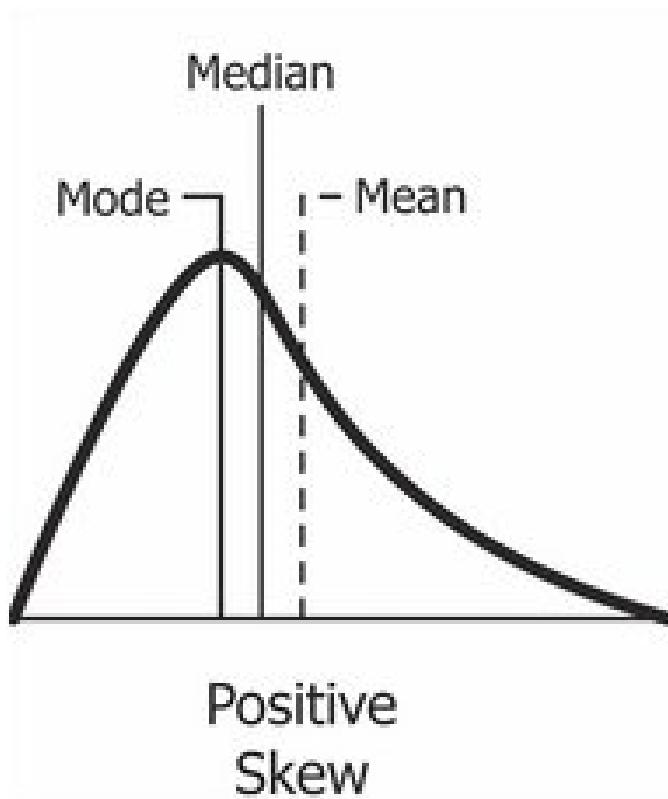
• Mean or Median imputation



- If the variable is normally distributed the mean and median are approximately the same



• Mean or Median imputation



- If the variable is skewed, the median is a better representation



• Mean / Median imputation: Assumptions

- Data is missing at random
- The missing observations, most likely look like the majority of the observations in the variable (aka, the mean / median)



• Mean / Median imputation: Advantages

- Easy to implement
- Fast way of obtaining complete datasets
- Can be integrated in production (during model deployment)



• Mean / Median imputation: Limitations

- Distortion of the original variable distribution
- Distortion of the original variance
- Distortion of the covariance with the remaining variables of the dataset
- **The higher the percentage of NA, the higher the distortions**



When to use Mean / Median Imputation

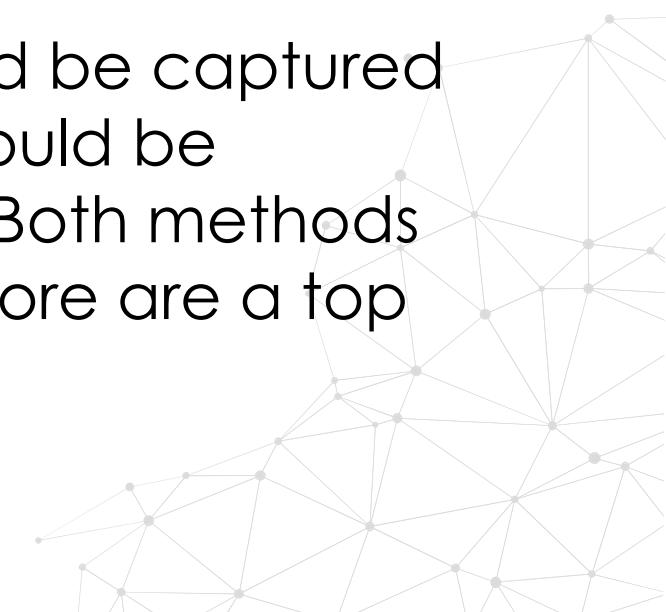
- Data is missing completely at random
- No more than 5% of the variable contains missing data



When to use Mean / Median Imputation

Typically, mean / median imputation is done together with adding a binary "missing indicator" variable to capture those observations where the data was missing (see lecture "Missing Indicator"), thus covering 2 angles:

if the data was missing completely at random, this would be captured by the mean /median imputation, and if it wasn't this would be captured by the additional "missing indicator" variable. Both methods are extremely straight forward to implement, and therefore are a top choice in data science competitions.



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
 - Mean / median imputation with pandas
 - Effect of the imputation on:
 - Variable distribution - variance
 - Interaction with other variables - covariance
 - Outliers



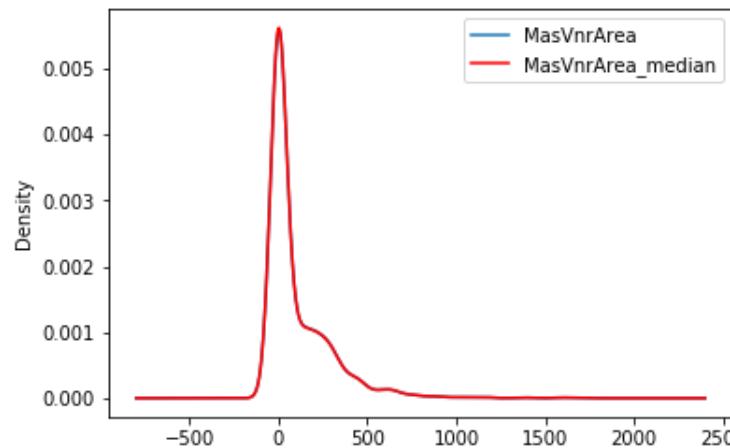
• Mean / Median Imputation

- The mean or median value should be calculated only in the train set and used to replace NA in both train and test sets.
- To avoid over-fitting



• Mean / Median Imputation effects

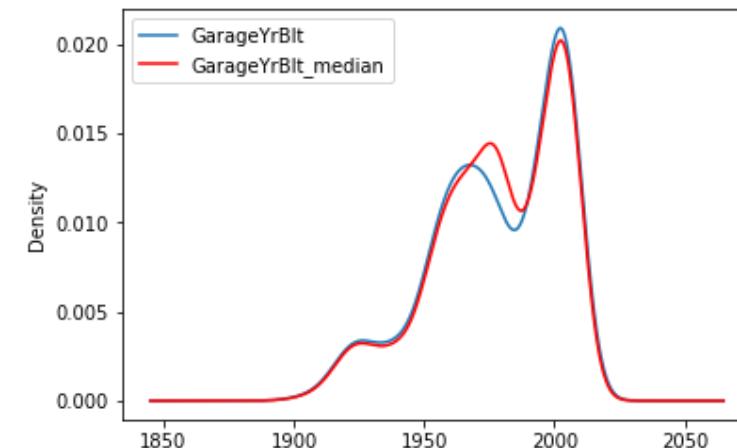
MasVnrArea 0.5% missing obs



Variance: 32983

Variance after imputation: 32874

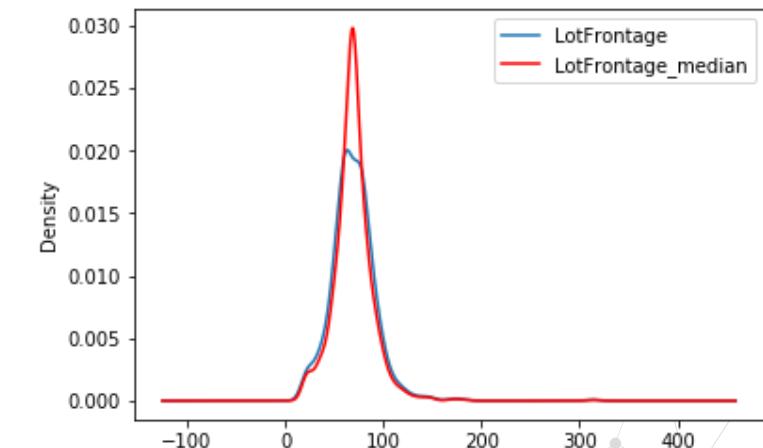
GarageYrBlt 5.5% missing obs



Variance: 624

Variance after imputation: 591

LotFrontage 17% missing obs

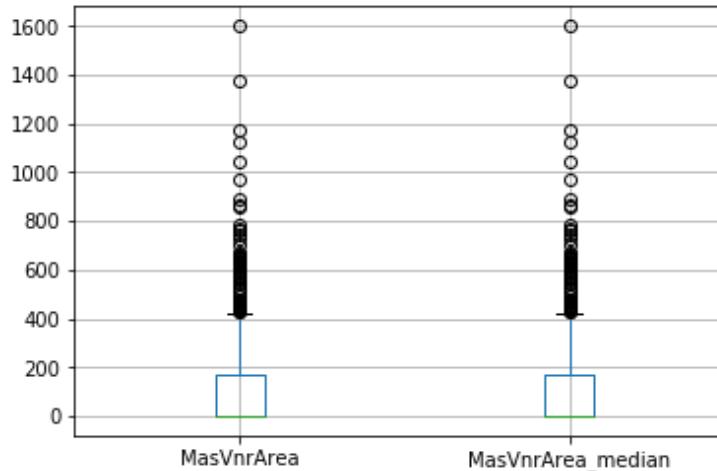


Variance: 532

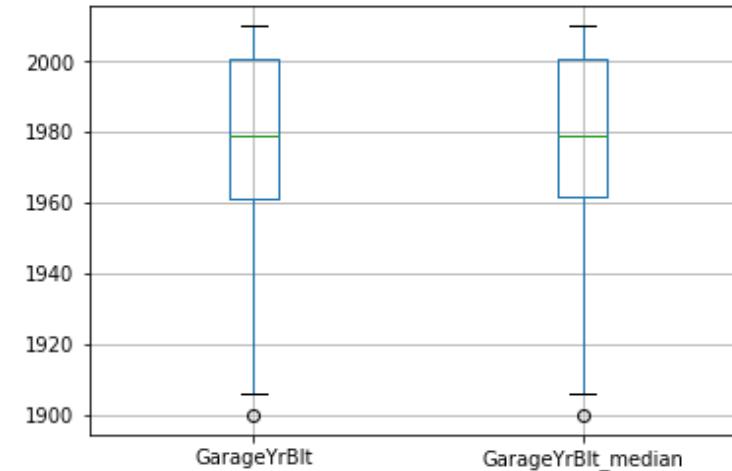
Variance after imputation: 434

• Mean / Median Imputation effects

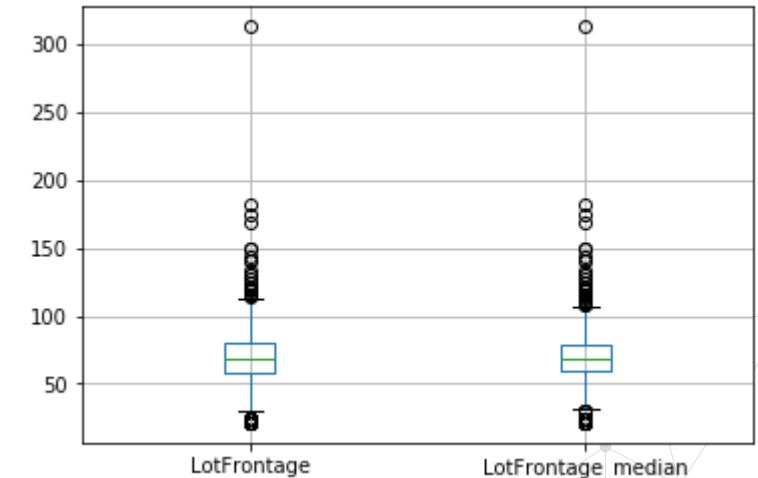
MasVnrArea 0.5% missing obs



GarageYrBlt 5.5% missing obs



LotFrontage 17% missing obs



More apparent outliers at
both ends of the distribution

• Mean / Median Imputation effects

	LotFrontage	OverallQual	MasVnrArea	BsmtUnfSF	TotalBsmtSF	1stFlrSF	GrLivArea	GarageYrBlt	WoodDeckSF	SalePrice
LotFrontage	532.587202	6.587119	6.805603e+02	9.496573e+02	2.908856e+03	3.379794e+03	3.919952e+03	30.611717	1.347414e+02	6.689645e+05
OverallQual	6.587119	1.843859	1.014970e+02	1.746147e+02	2.886241e+02	2.242973e+02	4.091242e+02	17.902809	3.168557e+01	8.320132e+04
MasVnrArea	680.560330	101.496976	3.298354e+04	7.540788e+03	2.478877e+04	2.086595e+04	3.520785e+04	1203.583792	3.208924e+03	6.836439e+06
BsmtUnfSF	949.657293	174.614725	7.540788e+03	1.875241e+05	7.513307e+04	4.987449e+04	5.203392e+04	1823.065167	-1.833201e+03	6.833028e+06
TotalBsmtSF	2908.855504	288.624075	2.478877e+04	7.513307e+04	1.682931e+05	1.212079e+05	8.615192e+04	3173.042442	1.227966e+04	2.003928e+07
1stFlrSF	3379.793504	224.297266	2.086595e+04	4.987449e+04	1.212079e+05	1.398656e+05	1.044401e+05	2009.195552	1.109406e+04	1.783631e+07
GrLivArea	3919.951834	409.124216	3.520785e+04	5.203392e+04	8.615192e+04	1.044401e+05	2.681277e+05	2738.982988	1.558395e+04	2.934477e+07
GarageYrBlt	30.611717	17.902809	1.203584e+03	1.823065e+03	3.173042e+03	2.009196e+03	2.738983e+03	624.305948	6.658911e+02	9.309355e+05
WoodDeckSF	134.741376	31.685571	3.208924e+03	-1.833201e+03	1.227966e+04	1.109406e+04	1.558395e+04	665.891118	1.648582e+04	3.029981e+06
SalePrice	668964.454191	83201.317781	6.836439e+06	6.833028e+06	2.003928e+07	1.783631e+07	2.934477e+07	930935.489321	3.029981e+06	6.105731e+09
LotFrontage_median	532.587202	5.384774	5.539213e+02	7.880954e+02	2.370929e+03	2.750747e+03	3.189686e+03	24.755173	1.060091e+02	5.448388e+05
MasVnrArea_median	674.423263	100.533003	3.298354e+04	7.472110e+03	2.465436e+04	2.080136e+04	3.496714e+04	1182.673336	3.212101e+03	6.790442e+06
GarageYrBlt_median	28.095264	16.875386	1.134381e+03	1.724142e+03	2.989473e+03	1.890272e+03	2.576346e+03	624.305948	6.276246e+02	8.774854e+05



THANK YOU

www.trainindata.com

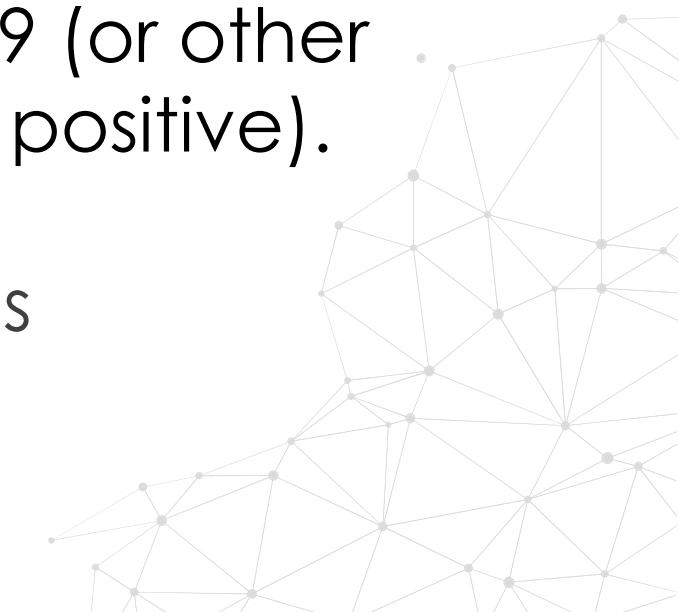




Arbitrary Value Imputation

• Arbitrary value imputation: definition

- Arbitrary value imputation consists of replacing all occurrences of missing values (NA) within a variable by an arbitrary value.
- Typically used arbitrary values are 0, 999, -999 (or other combinations of 9s) or -1 (if the distribution is positive).
- Suitable numerical and categorical variables
 - Categorical → “Missing”



• Arbitrary value imputation: example

Price
100
90
50
40
20
100
60
120
200

Arbitrary = 999



Price
100
90
50
40
20
100
999
60
120
999
200



• Arbitrary value imputation: example



The diagram illustrates the process of arbitrary value imputation. It consists of two tables of data, separated by a large blue arrow pointing from left to right. A large red 'X' is placed over the word 'Arbitrary' in the first table, indicating that this method is incorrect or undesirable.

Left Table (Original Data):

Price
100
90
50
40
20
100
60
120
200

Middle Text: Arbitrary = 99

Right Table (Imputed Data):

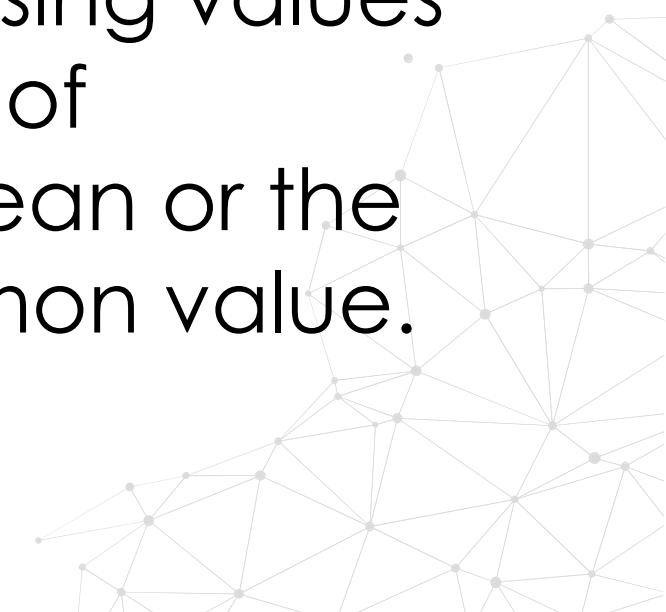
Price
100
90
50
40
20
100
999
60
120
999
200



• Arbitrary value imputation: Assumptions

Data is not missing at random.

If this is the case, we want to flag the missing values with a different (arbitrary) value, instead of replacing those occurrences with the mean or the median, which represent the most common value.



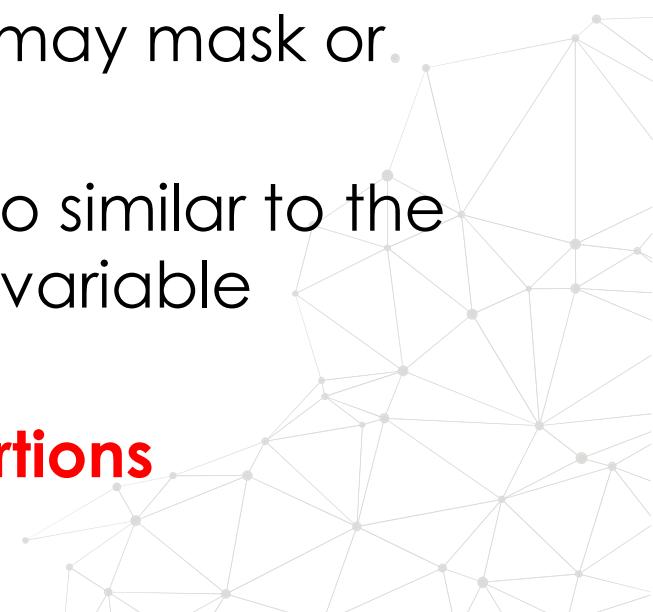
• Mean / Median imputation: Advantages

- Easy to implement
- Fast way of obtaining complete datasets
- Can be integrated in production (during model deployment)
- Captures the importance of being "missing" if there is one



• Mean / Median imputation: Limitations

- Distortion of the original variable distribution
- Distortion of the original variance
- Distortion of the covariance with the remaining variables of the dataset
- If the arbitrary value is at the end of the distribution it may mask or. create outliers
- Need to be careful not to chose an arbitrary value too similar to the mean or median (or any other common value of the variable distribution)
- **The higher the percentage of NA, the higher the distortions**



When to use arbitrary value imputation

- Data are not missing at random



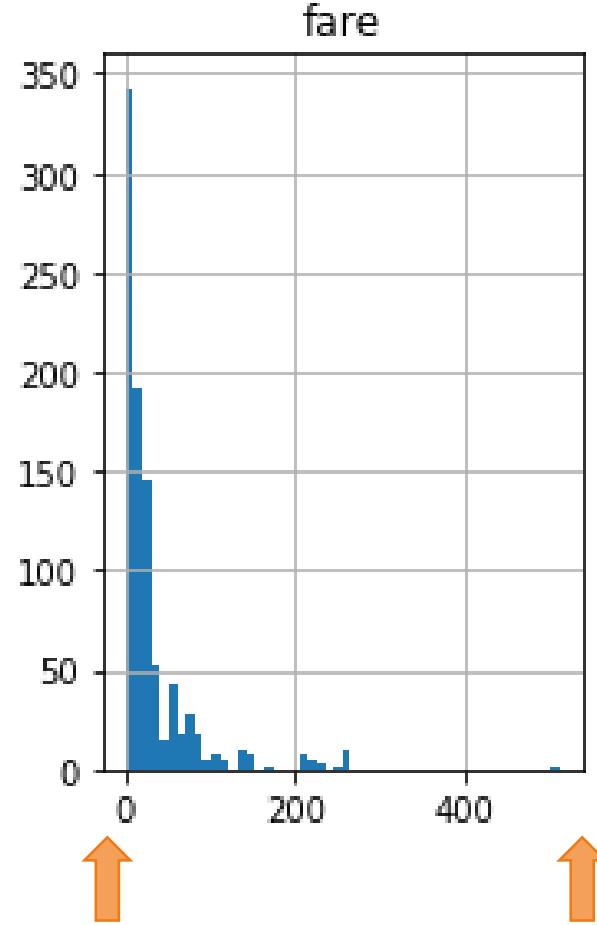
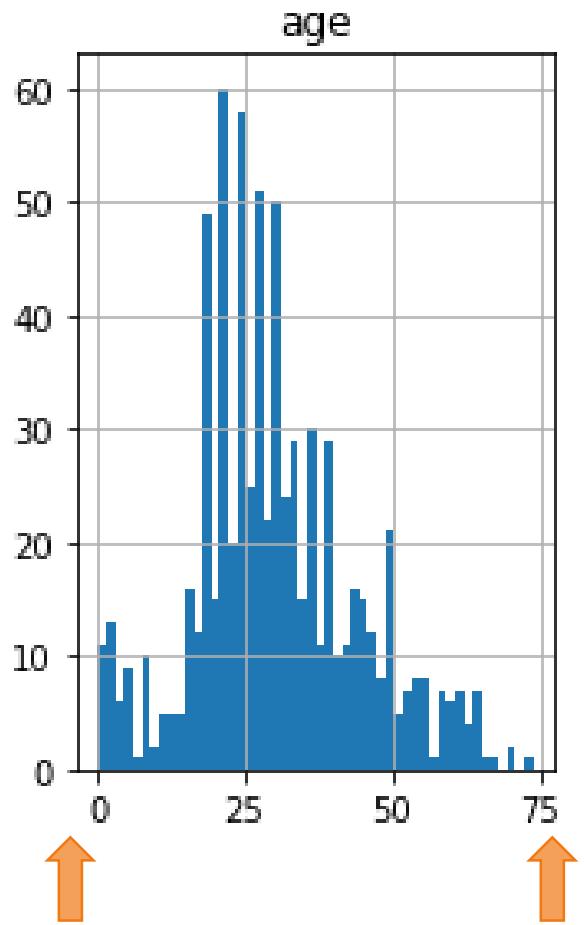
• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
 - Arbitrary value imputation with pandas
 - Effect of the imputation on:
 - Variable distribution - variance
 - Interaction with other variables - covariance
 - Outliers



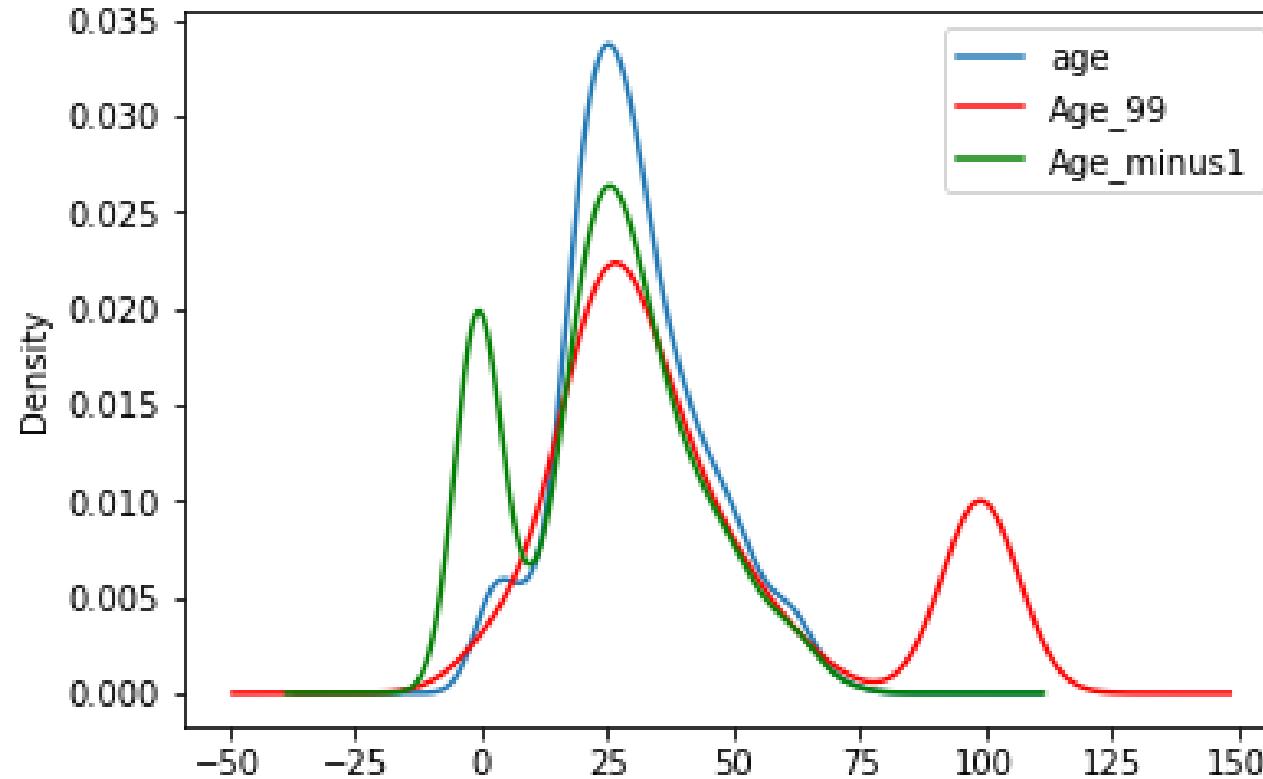
• Which arbitrary value to use?



Let's compare the effect of
using 99 or -1 for Age



• Arbitrary value imputation and distribution

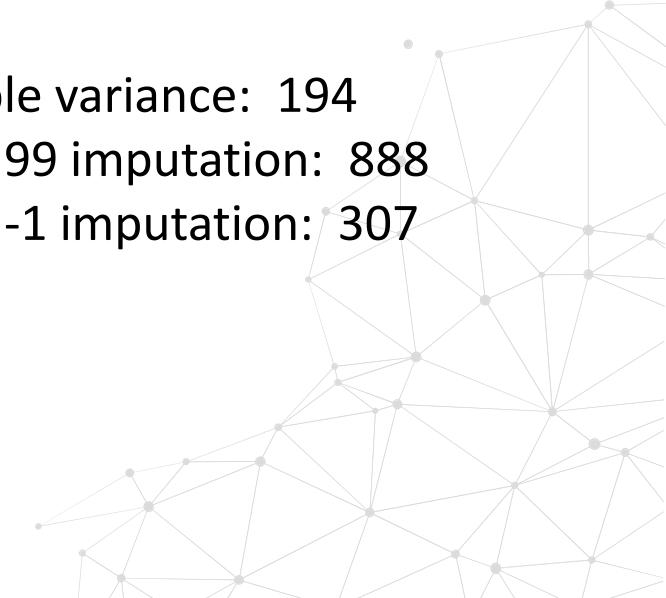


- ~20% of data is missing in Age

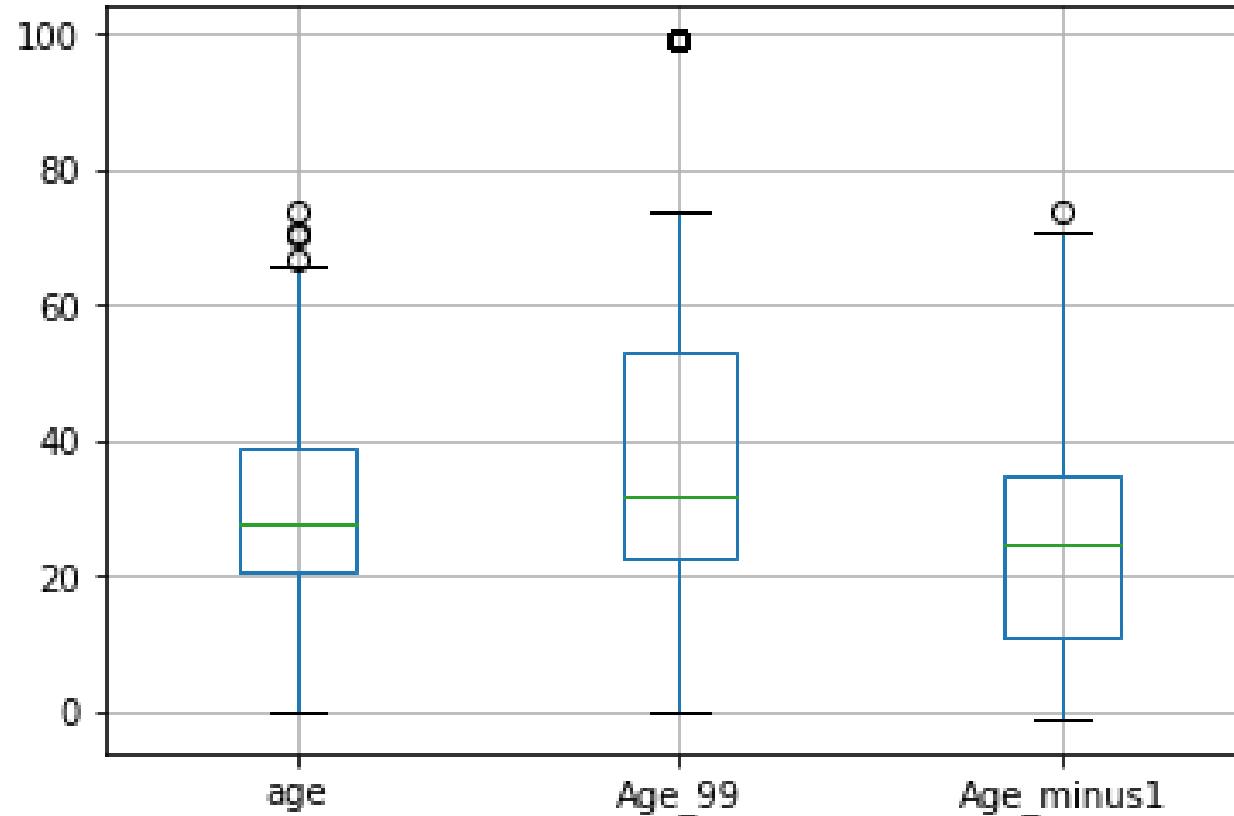
Original variable variance: 194

Variance after 99 imputation: 888

Variance after -1 imputation: 307



• Arbitrary value imputation and outliers



Masks outliers



• Arbitrary value imputation: effects

fare	
fare	2248.326729
age	136.176223
Age_99	-38.722001
Age_minus1	177.733891





THANK YOU

www.trainindata.com

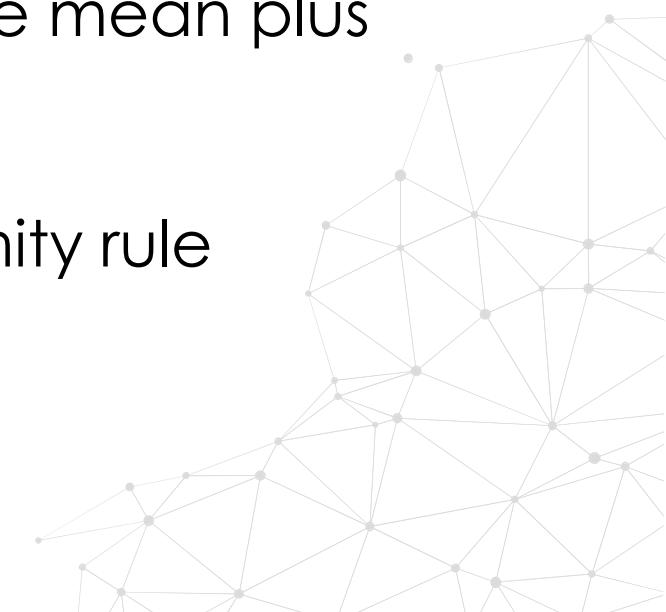




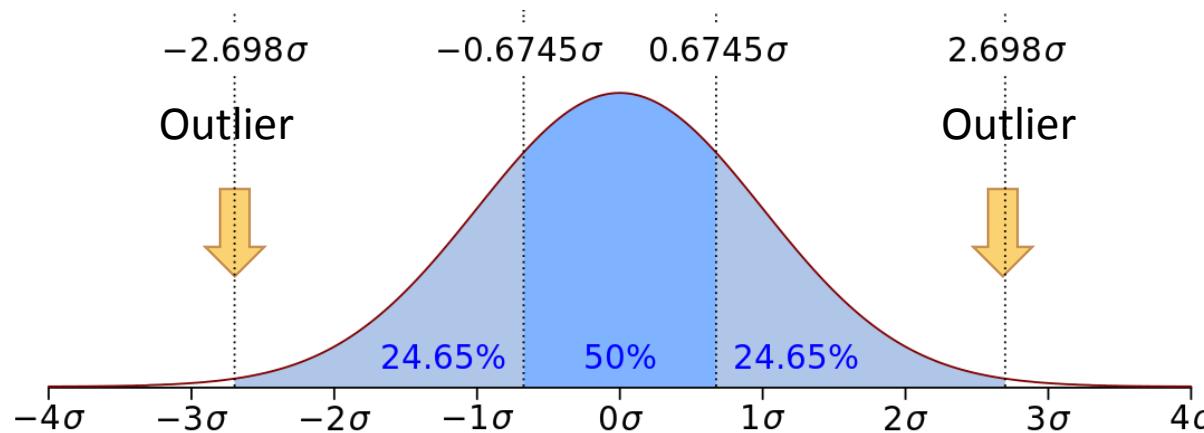
End of Tail Imputation

• End of tail imputation: definition

- End of tail imputation is equivalent to arbitrary value imputation, but automatically selecting arbitrary values at the end of the variable distributions.
- If the variable is normally distributed, we can use the mean plus or minus 3 times the standard deviation
- If the variable is skewed, we can use the IQR proximity rule
- Suitable numerical variables



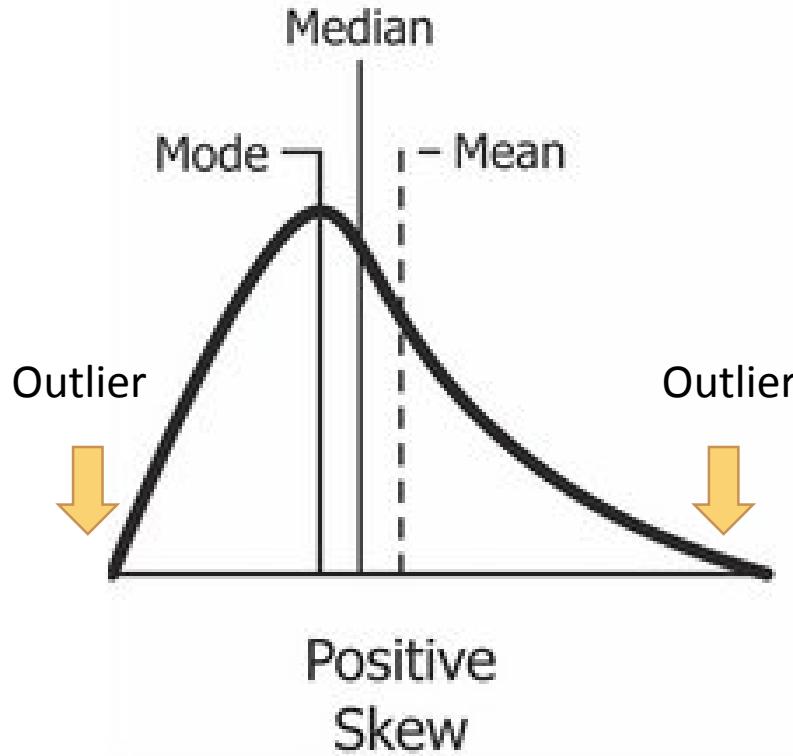
Normal distribution



- ~99% of the observations of a normally distributed variable lie within the mean $\pm 3 \times$ standard deviations.
- Values outside mean $\pm 3 \times$ standard deviations are considered outliers

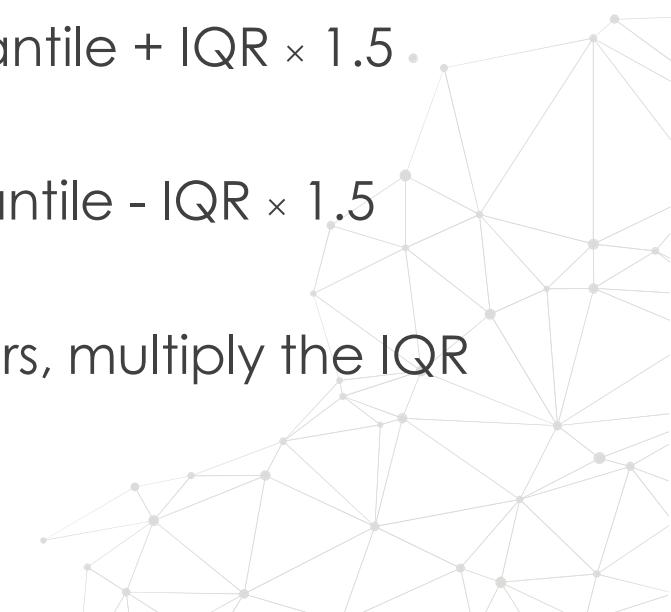


Skewed distributions



- The general approach is to calculate the quantiles, and then the inter-quantile range (IQR), as follows:
- $IQR = 75^{\text{th}} \text{ Quantile} - 25^{\text{th}} \text{ Quantile}$
- $\text{Upper limit} = 75^{\text{th}} \text{ Quantile} + IQR \times 1.5$.
- $\text{Lower limit} = 25^{\text{th}} \text{ Quantile} - IQR \times 1.5$

Note, for extreme outliers, multiply the IQR by 3 instead of 1.5



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
 - End of tail imputation with pandas
 - Effect of the imputation on:
 - Variable distribution - variance
 - Interaction with other variables - covariance
 - Outliers



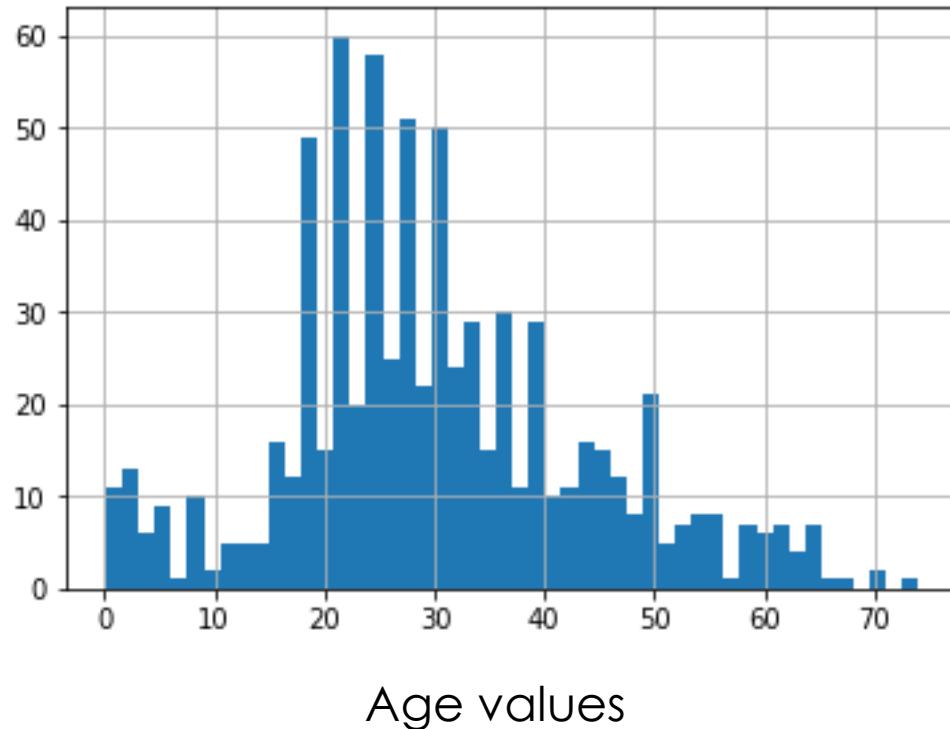
End of tail imputation: how to do it

- The values to replace missing data should be calculated only on the train set
- We need to divide the data set into train and test before doing the imputation techniques



End of tail imputation: how to do it

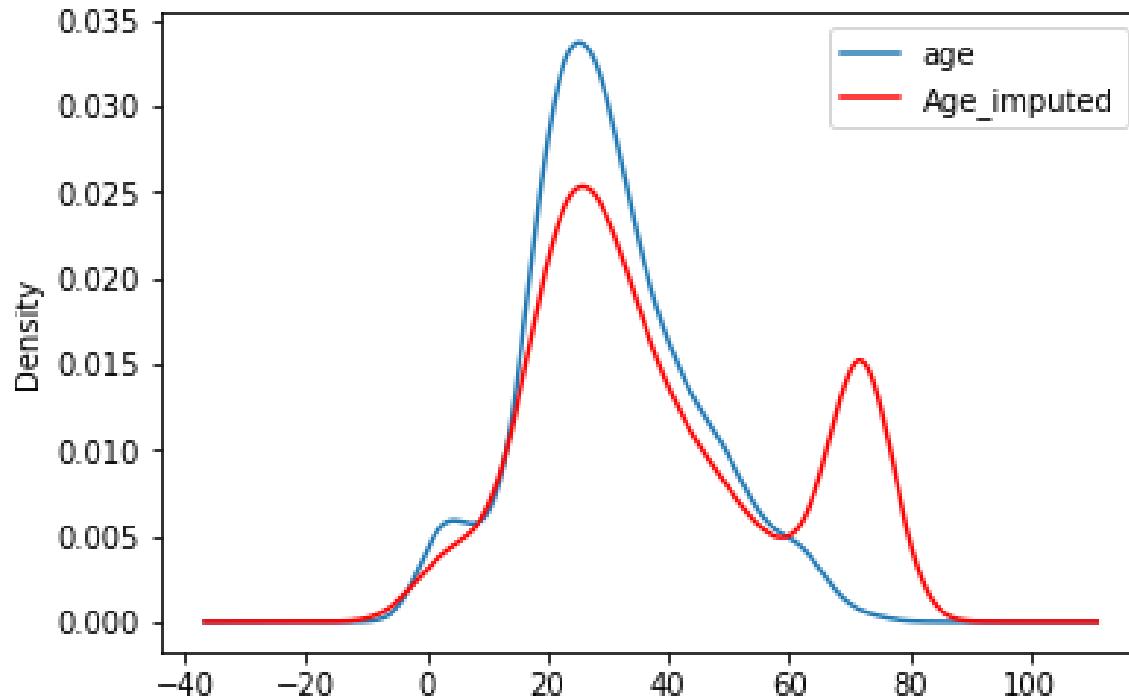
Histogram of Age from Titanic



$$\text{Mean}(\text{Age}) + 3 \times \text{std}(\text{Age}) = 72$$

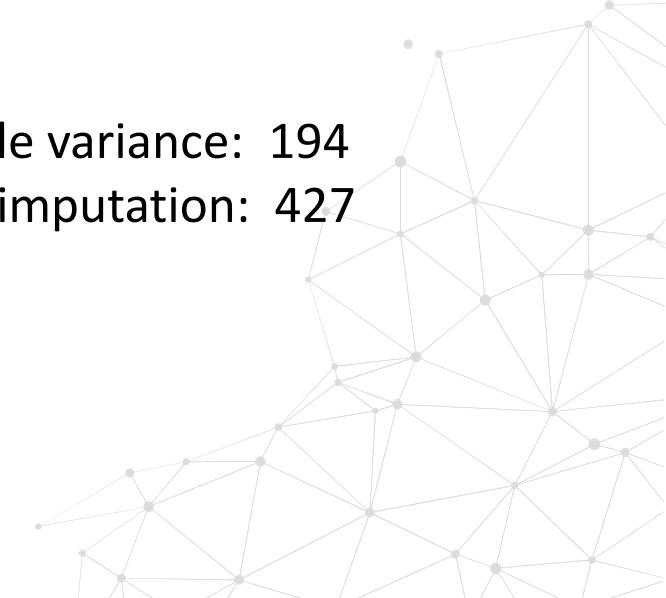


End of tail imputation and distribution

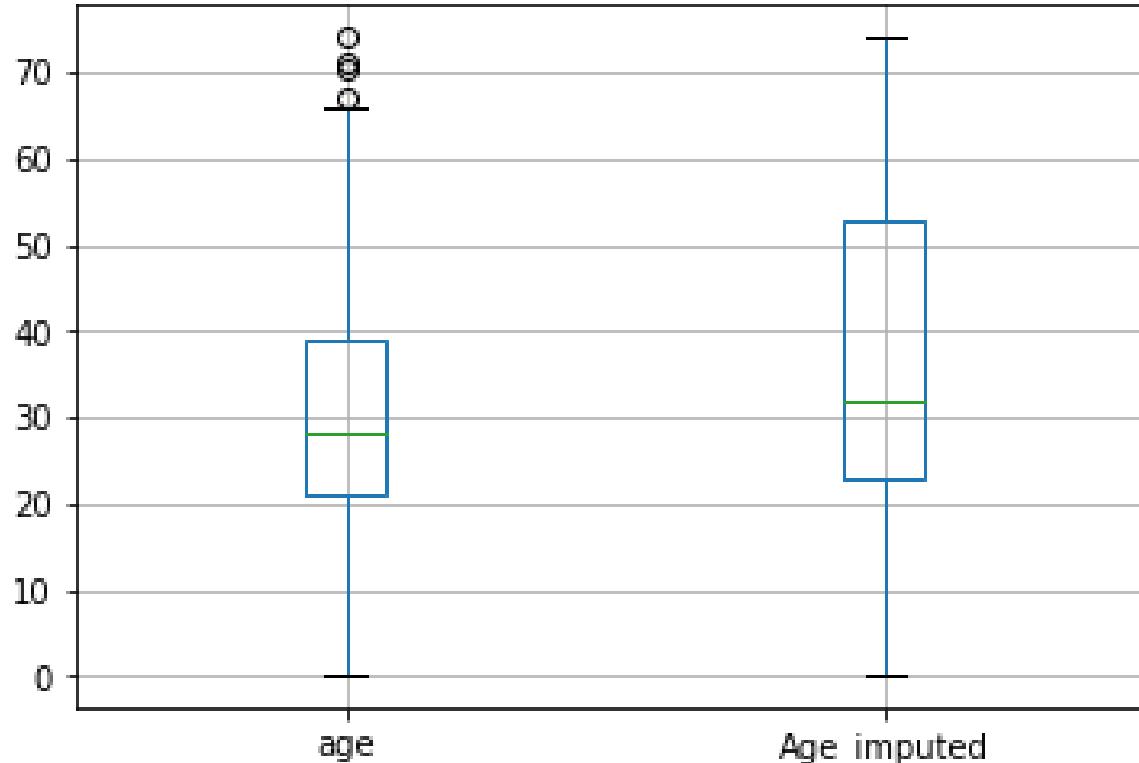


- ~20% of data is missing in Age

Original variable variance: 194
Variance after imputation: 427



End of tail imputation and outliers



Masks outliers



End of tail imputation: effects

fare	
fare	2248.326729
age	136.176223
Age_imputed	19.647139





THANK YOU

www.trainindata.com





Frequent Category Imputation

• Frequent Category imputation: definition

- **Mode** imputation consists of replacing all occurrences of missing values (NA) within a variable by the mode, or the **most frequent value**.
- Suitable numerical and categorical variables.
- In practice, we use this technique with categorical variables.



• Mode imputation: example

Make
Ford
Ford
Fiat
BMW
Ford
Kia
Fiat
Ford
Kia

Mode = Ford



Price
Ford
Ford
Fiat
BMW
Ford
Kia
Ford
Fiat
Ford
Ford
Kia



• Mode imputation: Assumptions

- Data is missing at random
- The missing observations, most likely look like the majority of the observations (aka, the mode)



• Mode imputation: Advantages

- Easy to implement
- Fast way of obtaining complete datasets
- Can be integrated in production (during model deployment)



• Mode imputation: Limitations

- Distortion the relation of the most frequent label with other variables within the dataset
- May lead to an over-representation of the most frequent label if there is a big number of NA
- **The higher the percentage of NA, the higher the distortions**



When to use Mode Imputation

- Data is missing completely at random
- No more than 5% of the variable contains missing data



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
 - Frequent category imputation with pandas
 - Effect of the imputation on:
 - Variable distribution - proportions
 - Interaction with other variables - target





THANK YOU

www.trainindata.com





Missing Category Imputation

Missing Category imputation: definition

- This method consists in treating missing data as an additional label or category of the variable.
 - Missing observations are grouped in the newly created label '**Missing**'.
- This is the most widely used method of missing data imputation for categorical variables.
- Suitable for categorical variables



Missing Category imputation: example

Make
Ford
Ford
Fiat
BMW
Ford
Kia
Fiat
Ford
Kia

“Missing”



Price
Ford
Ford
Fiat
BMW
Ford
Kia
Missing
Fiat
Ford
Missing
Kia



Missing Category imputation: Advantages

- Easy to implement
- Fast way of obtaining complete datasets
- Can be integrated in production (during model deployment)
- Captures the importance of "missingness" if there is one
- No assumption made on the data



Missing Category imputation: Limitations

- If the number of NA is small creating an additional category is in essence adding another rare label to the variable.



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
 - Missing category imputation with pandas
 - Effect of the imputation on:
 - Variable distribution - proportions
 - Interaction with other variables - target





THANK YOU

www.trainindata.com





Random Sample Imputation

• Random sample imputation: definition

- Random sampling consist in taking a random observation from the pool of available observations of the variable, and using that randomly extracted value to fill the NA.
- Suitable for both numerical and categorical variables



Random sample imputation: example

Price	Make
100	Ford
90	Ford
50	Fiat
40	BMW
20	Ford
100	
	Kia
60	Ford
120	BMW
200	Kia

Random Sample



Price	Make
100	Ford
90	Ford
50	Fiat
40	BMW
20	Ford
100	
	Kia
100	Ford
100	Kia
60	Ford
120	BMW
90	Kia
200	Kia



• Random sample imputation: Assumptions

- Data is missing at random
- The idea is to replace the population of missing values with a population of values with the same distribution of the original variable.



• Random sample imputation: Advantages

- Easy to implement
- Fast way of obtaining complete datasets
- Can be integrated in production (during model deployment)
- Preserves the variance of the variable



• Random sample imputation: Limitations

- Randomness
- The relationship of imputed variables with other variables may be affected if there are a lot of NA
- Memory heavy for deployment, as we need to store the original training set to extract values from and replace the NA in coming observations.



Random sample imputation: Randomness

Price	Make
100	Ford
90	Ford
50	Fiat
40	BMW
20	Ford
100	
	Kia
60	Ford
120	BMW
200	Kia

Random Sample 1

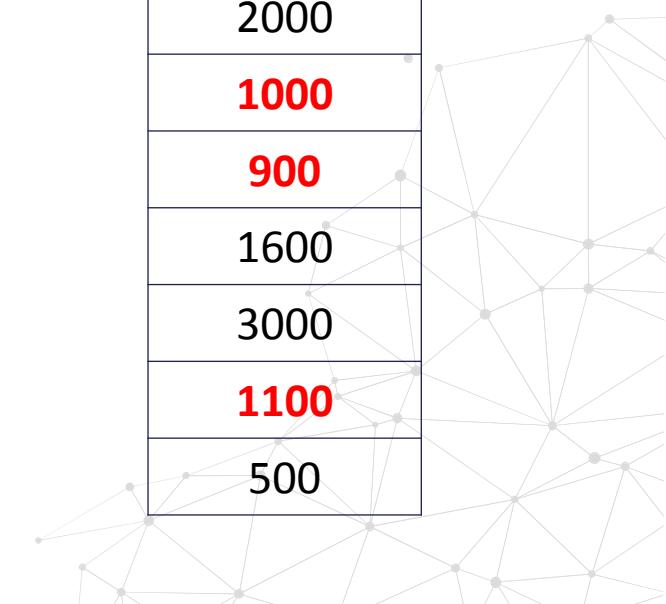


Price	Make
100	Ford
90	Ford
50	Fiat
40	BMW
20	Ford
100	Ford
100	Kia
60	Ford
120	BMW
90	Kia
200	Kia

Prediction 1



Prediction
1000
1200
500
4000
2000
1000
900
1600
3000
1100
500



Random sample imputation: Randomness

Price	Make
100	Ford
90	Ford
50	Fiat
40	BMW
20	Ford
100	
	Kia
60	Ford
120	BMW
200	Kia

Random Sample 2

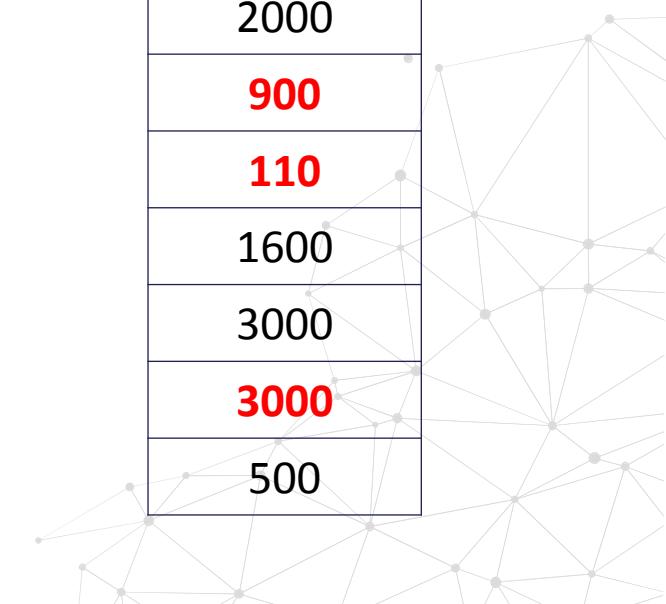


Price	Make
100	Ford
90	Ford
50	Fiat
40	BMW
20	Ford
100	
90	Kia
60	Ford
120	BMW
120	BMW
200	Kia

Prediction 2



Prediction
1000
1200
500
4000
2000
900
110
1600
3000
3000
500



Random sample imputation: Randomness

Price	Make
100	Ford
90	Ford
50	Fiat
40	BMW
20	Ford
100	
	Kia
60	Ford
120	BMW
200	Kia

Random Sample 3

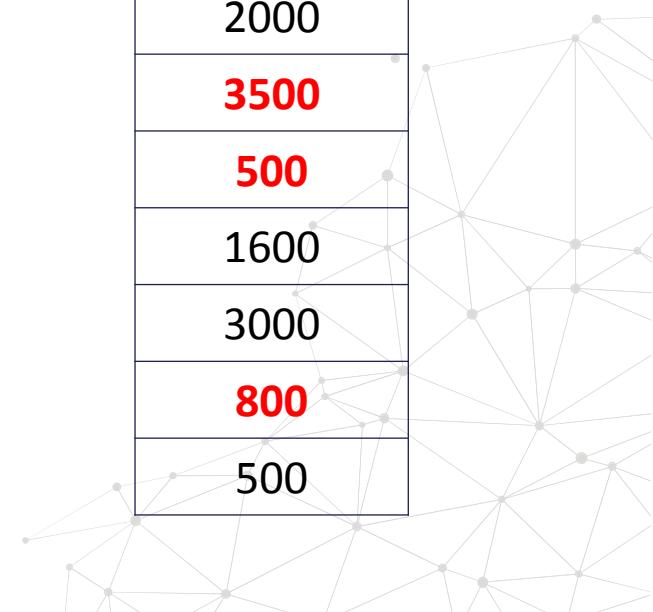


Price	Make
100	Ford
90	Ford
50	Fiat
40	BMW
20	Ford
100	BMW
200	Kia
60	Ford
120	BMW
120	Ford
200	Kia

Prediction 3



Prediction
1000
1200
500
4000
2000
3500
500
1600
3000
800
500



• Random sample imputation: Randomness

- Every time we score the same observation, we may obtain a different prediction
- Unwanted side-effect
- **Set the seed using other variables in the dataset**



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
 - Random Sample imputation with pandas
 - Effect of the imputation on:
 - Variable distribution
 - Outliers



Random sample Imputation

- The population of values used to replace NA should be the train set.
- To avoid over-fitting





THANK YOU

www.trainindata.com





Train In Data

Missing Indicator

Missing indicator: definition

- A Missing Indicator is an additional binary variable, which indicates whether the data was missing for an observation (1) or not (0).
- Suitable for numerical and categorical variables



Missing indicator: example

Price
100
90
50
40
20
100
60
120
200

Missing Indicator



Price	MI
100	0
90	0
50	0
40	0
20	0
100	0
	1
60	0
120	0
	1
200	0



Missing indicator + Mean Imputation

Price
100
90
50
40
20
100
60
120
200

Mean = 86.66



Price	MI
100	0
90	0
50	0
40	0
20	0
100	0
86.66	1
60	0
120	0
86.66	1
200	0



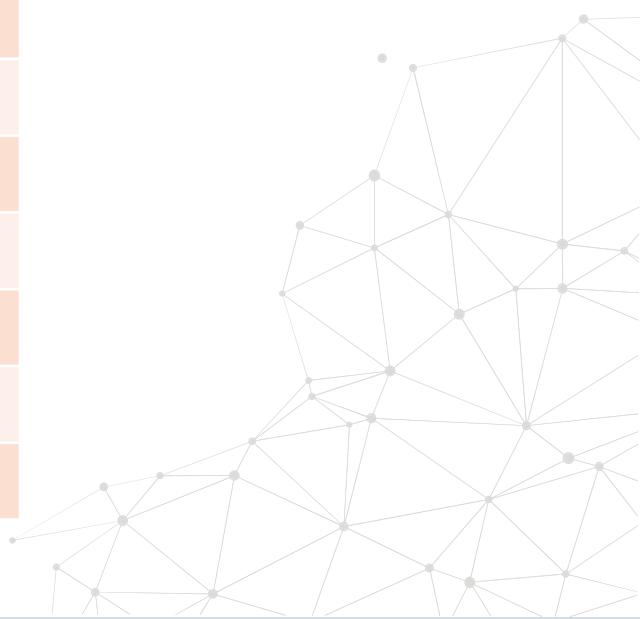
Missing indicator: example

Make
Ford
Ford
Fiat
BMW
Ford
Kia
Ford
BMW
Kia

Missing Indicator



Make	MI
Ford	0
Ford	0
Fiat	0
BMW	0
Ford	0
	1
Kia	0
Ford	0
BMW	0
	1
Kia	0



Missing indicator + Frequent Category

Make
Ford
Ford
Fiat
BMW
Ford
Kia
Ford
BMW
Kia

Frequent category = Ford



Make	MI
Ford	0
Ford	0
Fiat	0
BMW	0
Ford	0
Kia	0
Ford	0
BMW	0
Kia	0



Missing indicator: use

- The Missing Indicator is used together with methods that assume data is missing at random:
 - Mean, median, mode imputation
 - Random sample imputation



Missing indicator: Assumptions

- Data is NOT missing at random
- Missing data are predictive



Missing indicator: Advantages

- Easy to implement
- Captures importance of missing data
- Can be integrated in production (during model deployment)



Missing indicator: Limitations

- Expands the feature space
- Original variable still needs to be imputed
- **Many missing indicators may end up being identical or very highly correlated**

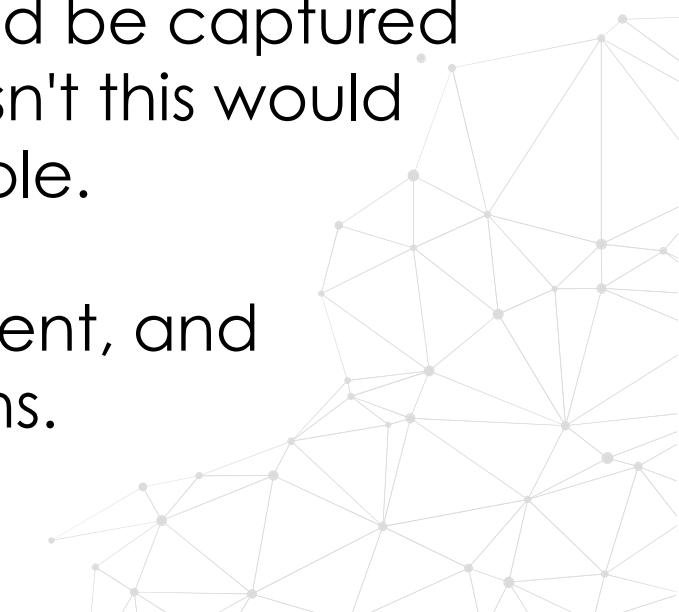


When to use a missing indicator

Typically, mean, median and mode imputation are done together with adding a binary "missing indicator" variable to capture those observations where the data was missing (see lecture "Missing Indicator"), thus covering 2 angles:

if the data was missing completely at random, this would be captured by the mean, median or mode imputation, and if it wasn't this would be captured by the additional "missing indicator" variable.

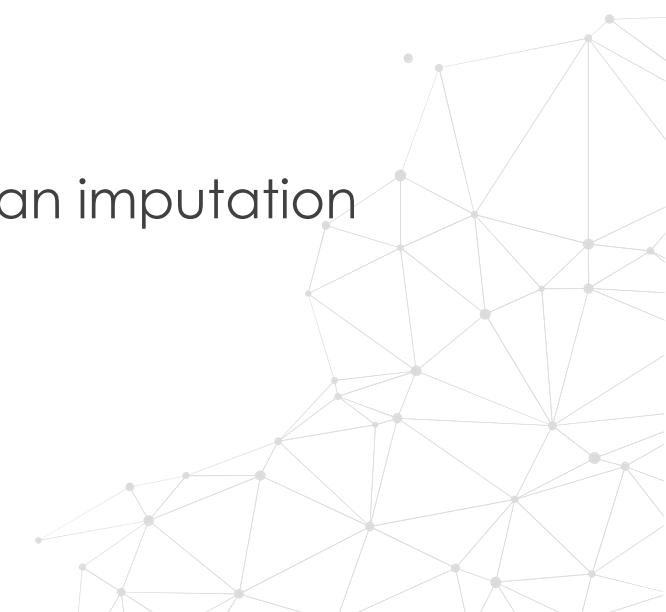
Both methods are extremely straight forward to implement, and therefore are a top choice in data science competitions.



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
 - Missing indicator with pandas and NumPy
 - Followed by median imputation



Missing indicator with NumPy

In [6]: ►

```
1 # add the missing indicator
2
3 # this is done very simply by using np.where from numpy
4 # and isnull from pandas:
5
6 X_train['Age_NA'] = np.where(X_train['age'].isnull(), 1, 0)
7 X_test['Age_NA'] = np.where(X_test['age'].isnull(), 1, 0)
8
9 X_train.head()
```

Out[6]:

	age	fare	Age_NA
501	13.0	19.5000	0
588	4.0	23.0000	0
402	30.0	13.8583	0
1193	NaN	7.7250	1
686	22.0	7.7250	0



Missing indicator + Median imputation

```
|: └──▶ 1 # for example median imputation  
2  
3 median = X_train['age'].median()  
4  
5 X_train['age'] = X_train['age'].fillna(median)  
6 X_test['age'] = X_test['age'].fillna(median)  
7  
8 # check that there are no more missing values  
9 X_train.isnull().mean()
```

```
t[9]: age      0.0  
       fare     0.0  
       Age_NA   0.0  
       dtype: float64
```

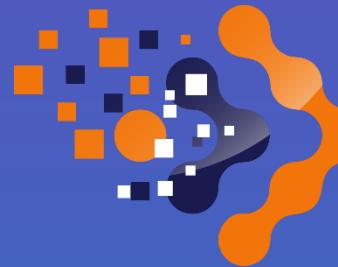




THANK YOU

www.trainindata.com



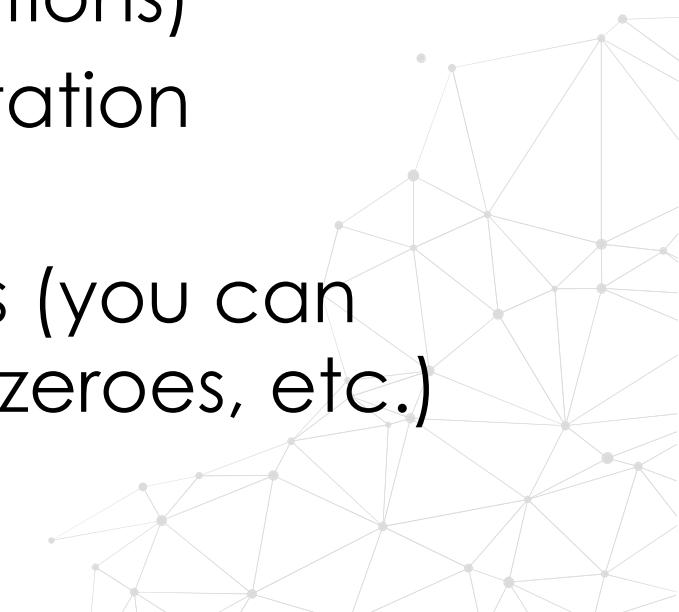


Train In Data

Missing Imputation with Scikit-learn: `SimpleImputer()`

• SimpleImputer(): Advantages

- Simple to use if applied to the entire dataframe
- Maintained by the Scikit-learn developers: good quality code
- Fast computation (it uses NumPy for calculations)
- Allows for grid search over the various imputation techniques
- Allows for different missing values encodings (you can indicate if the missing values are np.nan, or zeroes, etc.)



SimpleImputer(): Limitations

- Returns a NumPy array instead of a pandas dataframe, inconvenient for data analysis
- Needs to use additional classes to select which features to impute:
 - requires more lines of code
 - additional classes still in beta (may change without warning)
 - not so straightforward to use anymore.





THANK YOU

www.trainindata.com





Missing Data Imputation with Feature-engine

• Feature-engine



- <https://www.trainindata.com/feature-engine>
- <https://feature-engine.readthedocs.io/en/latest/>
- https://github.com/solegalli/feature_engine

pip install feature-engine



• Feature-engine: Advantages

- Feature-engine includes all the feature engineering techniques described in the course
- Feature-engine works like Scikit-learn, so it is easy to learn
- Feature-engine allows you to implement specific engineering steps to specific feature subsets
- Feature-engine can be integrated with the Scikit-learn pipeline allowing for smooth model building
- **Feature-engine allows you to design and store a feature engineering pipeline with bespoke procedures for different variable groups.**



Feature-engine: with fit and transform



- `fit()` → learns parameters from train set
- `transform()` → transforms data



• Feature-engine transforms specific variables

Feature engine allows you to specify variable groups easily



```
1 # Let's do mean imputation this time  
2 # and let's do it over 2 of the 3 numerical variables  
3  
4 imputer = mdi.MeanMedianImputer(imputation_method='mean',  
5                                 variables=['LotFrontage', 'MasVnrArea'])  
6  
7 imputer.fit(X_train)
```

```
MeanMedianImputer(imputation_method='mean',  
variables=['LotFrontage', 'MasVnrArea'])
```



• Feature-engine stores the transformation parameters



```
| 1 # now the imputer uses only the variables we indicated  
| 2  
| 3 imputer.variables  
  
['LotFrontage', 'MasVnrArea']
```

```
| 1 # and we can see the value assigned to each variable  
| 2 imputer.imputer_dict_  
  
{'LotFrontage': 69.66866746698679, 'MasVnrArea': 103.55358898721731}
```

Feature-engine: Limitations

- ✓ I would like to hear your feedback!!!





THANK YOU

www.trainindata.com





Multivariate Imputation

• Univariate Imputation

Imputes values in each feature using only non-missing values or statistical estimates derived from that same feature.

E.g.,

- Mean and median imputation
- Mode imputation
- Random Sample imputation
- Etc. – Section 4



Multivariate Imputation

Uses other features in the dataset to estimate the missing values.



Multivariate Imputation

Use neighbouring variables to predict the missing value.

- KNN
 - Regression
 - Any ML model we like



Multivariate Imputation Techniques

KNN Imputation

MICE (multivariate Imputation by Chained Equations)

MissForest – extends MICE using Random Forests



• Multivariate Imputation: Aim

Predict, as accurately as possible, the “**real**” values of those **missing observations**.

We are often not interested in those real values, instead in **making accurate predictions of our target**.



Multivariate Imputation: Considerations



Adds some complexity to the Machine Learning Pipeline





THANK YOU

www.trainindata.com





Train In Data

KNN Imputation

KNN Imputation

Determines the missing data point value, as the weighted average of the values of its K nearest neighbours.

The logic

If an observation looks very similar to other observations in the data set, most likely, the missing value would be similar to the value shown in those similar observations.



KNN Imputation: procedure

For each variable with missing value:

- 1- train a KNN using the other variables
 - 2- Find the K closest neighbours

KNN Imputation: procedure

For each variable with missing value:

- 1- train a KNN using the other variables
 - 2- Find the K closest neighbours

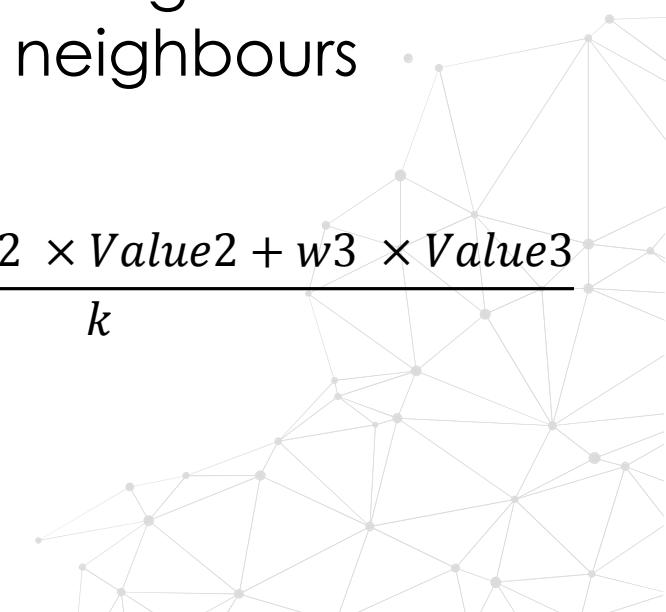
KNN Imputation: procedure

Var 1	Var 2	Var 3	Var 4	Var 5
	Value1			
	Value1			
	Value1			

For each variable with missing value:

3 – Determine the weighted average of the K neighbours

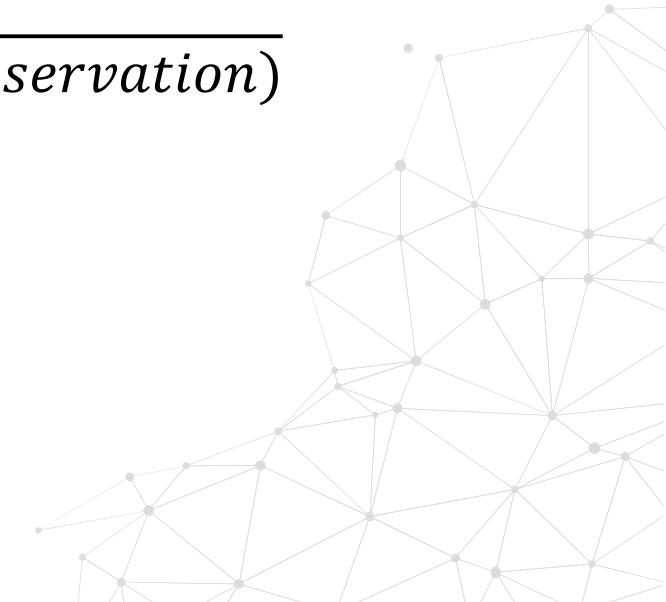
$$NA\ replacement = \frac{w_1 \times Value1 + w_2 \times Value2 + w_3 \times Value3}{k}$$



• KNN imputation: weight

Uniform: all neighbours count equally ($w_i = 1$)

Distance: $W_i = \frac{1}{\text{Euclidean distance(neighbour} - \text{observation})}$



• KNN imputation: challenge

Find the best number of neighbours → K

If you have time in your hands, this turns into a regression problem, where we need a training set to train the KNN and determine the optimal K

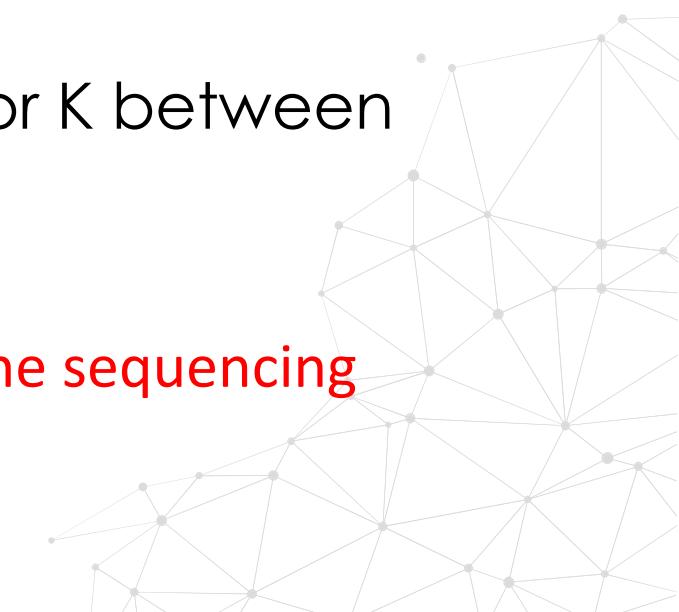


• KNN general guidelines

The authors from the algorithm claim:

- A small percentage of missing data makes the imputation more precise (up to 20% missing data)
- The method is relative insensitive to the value of K, for K between 10 and 20

These parameters were developed to fill in missing values in gene sequencing data and may not extend to other problems.





THANK YOU

www.trainindata.com





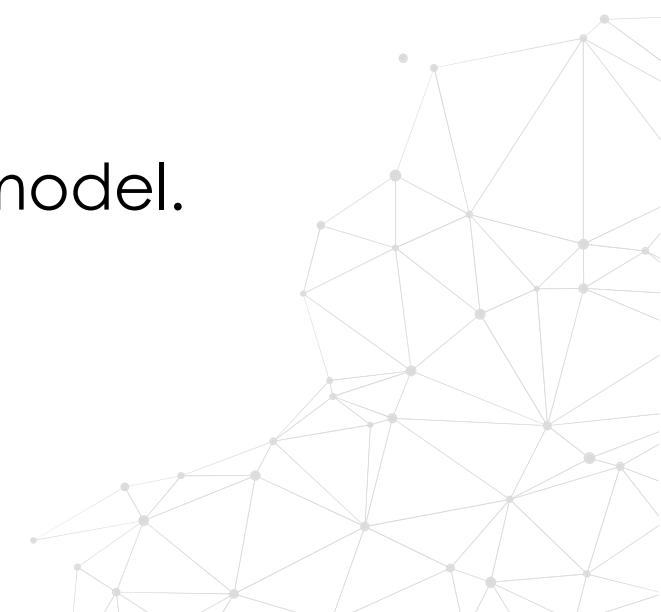
MICE

MICE

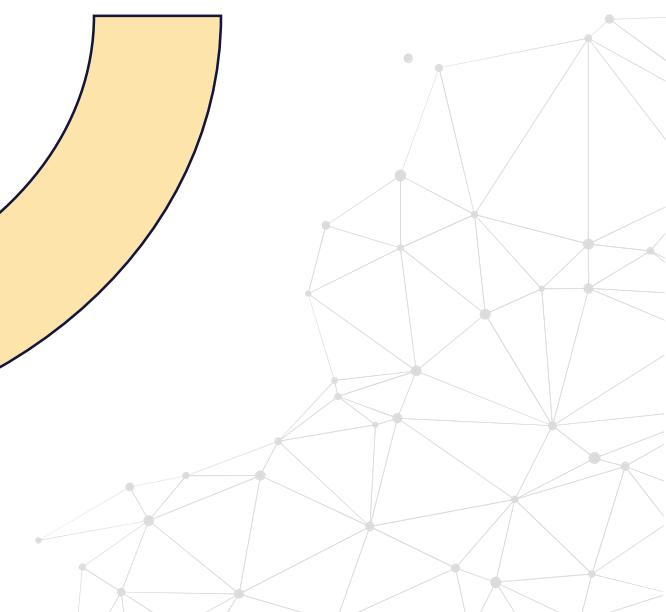
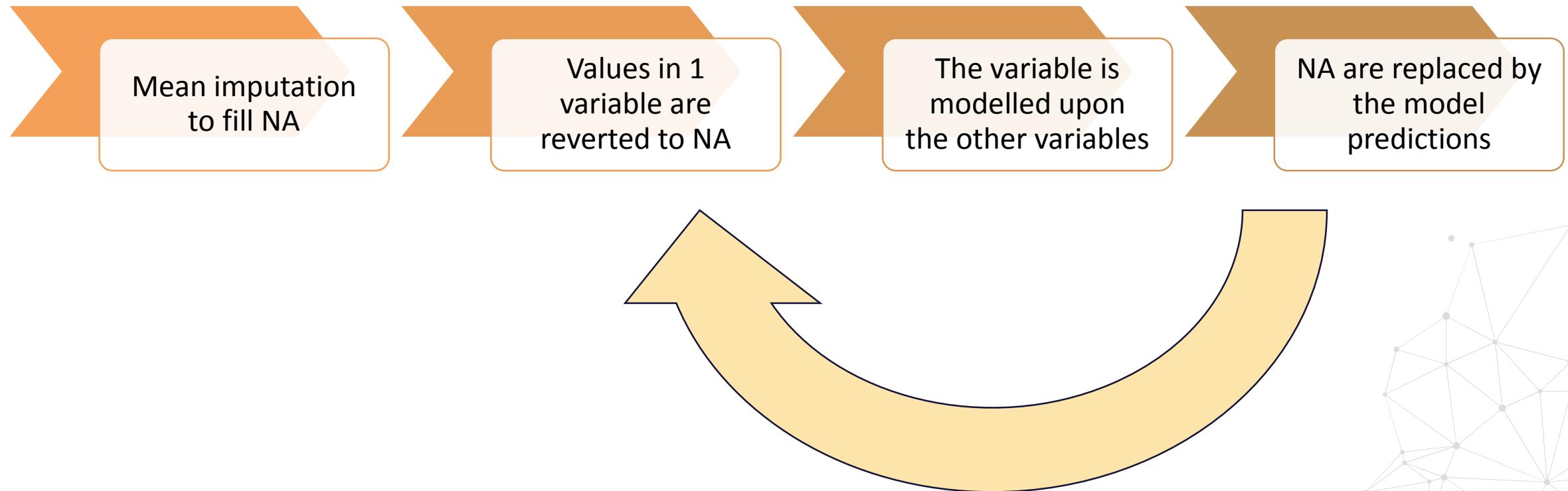
Multivariate Imputation of Chained Equations

A series of models whereby each variable is modelled conditional upon the other variables in the data.

Each incomplete variable is imputed by a separate model.



MICE: framework



• MICE: framework

After all variables with NA have been modelled based on the other variables, 1 round of imputation is completed.

The procedure repeats itself n times, usually 10 imputation cycles are enough to find stable parameters for the models.



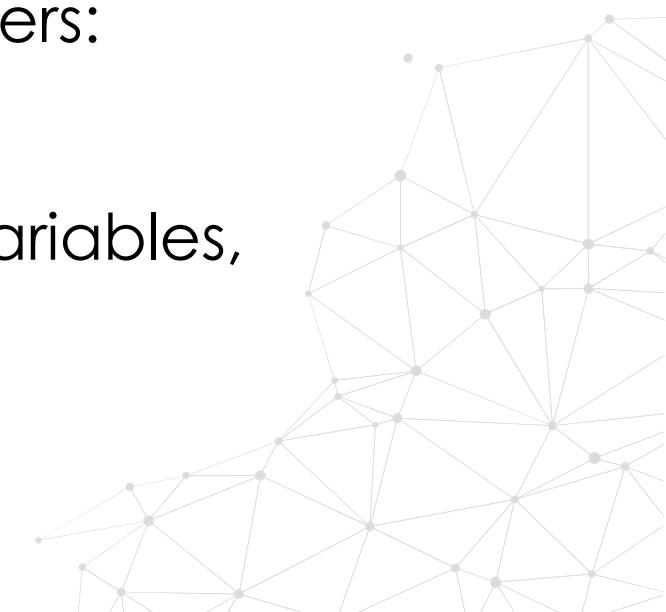
• MICE: why multiple rounds?

In the first round, we are modelling the variables based on the other ones, which themselves may contain NA.

- Thus, the predictions might be biased.

As we continue to regress one variable upon the others:

- We obtain better estimates for the NA,
- These estimates are used to regress the other variables,
- Thus returning more accurate predictions.



• MICE: assumptions

- Data is MAR
- The NA in the variables can be modelled by the other variables in the dataset, and does not depend on external sources.





MICE

Considerations



• MICE: variable relationship

Variables may have linear or non-linear relationships

- Find best model to predict the missing data, i.e., Linear Regression, Bayes, tree based algorithms, etc.
- Optimise the model parameters.



• MICE: variable nature

Depending on the nature of our variables, we should use different models.

- Binary variables should be modelled with classification algorithms
- Continuous variables should be modelled with regression algorithms
- Discrete variables should be modelled with Poisson

Not possible to automate with current tools. We would have to train each model manually.

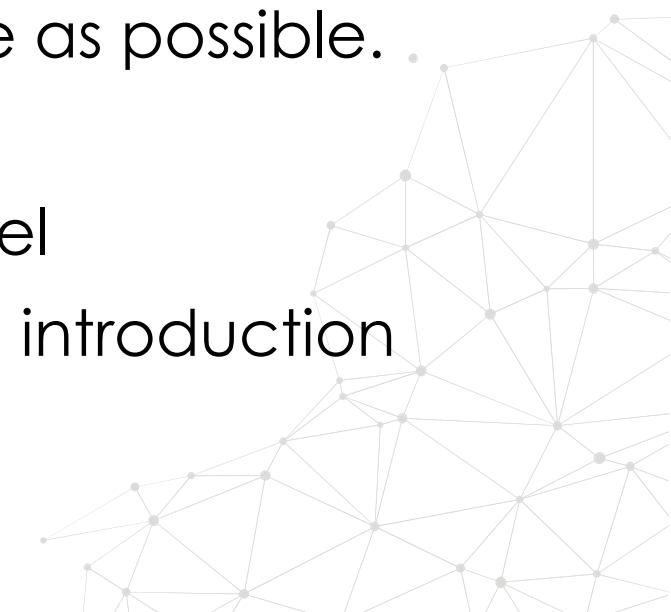


• MICE: Which variables should we use as predictors?

Authors suggest that using every available bit of available information yields multiple imputations that have minimal bias and maximal certainty.

→ the number of predictors in should be as large as possible.

- Include all variables that will be used in the final model
- Add variables thought to be somehow related to the introduction of missing data.



• MICE: more considerations

- “Circular” dependence can occur: same observations show NA on several variables → the variables may be correlated





THANK YOU

www.trainindata.com





missForest

missForest

MICE implementation where Random Forests are used to regress the variable with NA to the other variables in the data.

- Works well with mixed data types (categorical and continuous)
- Robust and accurate
- Handles non-linear relationships and variable interactions





THANK YOU

www.trainindata.com





Categorical Encoding

Categorical Encoding

- Categorical encoding refers to replacing the category strings by a numerical representation

The goal of categorical encoding is:

- To produce variables that can be used to train machine learning models
- To build predictive features from categories



Categorical Encoding Techniques

Traditional techniques

One hot encoding

Count / frequency encoding

Ordinal / Label encoding

Monotonic relationship

Ordered label encoding

Mean encoding

Weight of evidence

Alternative techniques

Binary encoding

Feature hashing

Others



Monotonic relationship

What is a monotonic relationship between the variable and the target?

- When variable increases and the target increases. Or,
- When variable increases and the target decreases.



Monotonic relationship

- Improve the performance of linear models
- May improve the performance of tree based models
 - Creates shallower trees
- Often, organisations want to include monotonic constraints
 - Insurance premiums decreasing with age



Encoding Techniques: Rare labels



Particularly important for model deployment

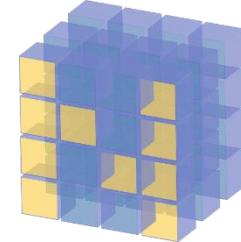
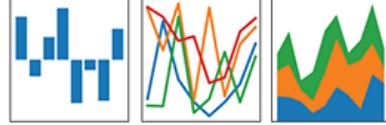
- ✓ One hot encoding of frequent categories
- ✓ Grouping of rare categories



Categorical Encoding Techniques

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



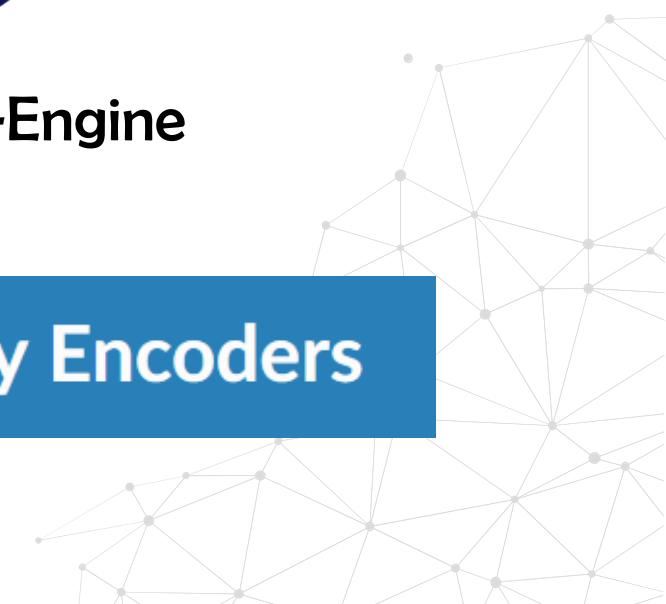
NumPy



Feature-Engine



Category Encoders



Objectives

Understand the different techniques for categorical encoding.



Learn multiple techniques



Understand their impact on the variable and
the machine learning model



Learn how to implement it with pandas, Scikit-learn, and
Feature-Engine, within a machine learning pipeline



Content



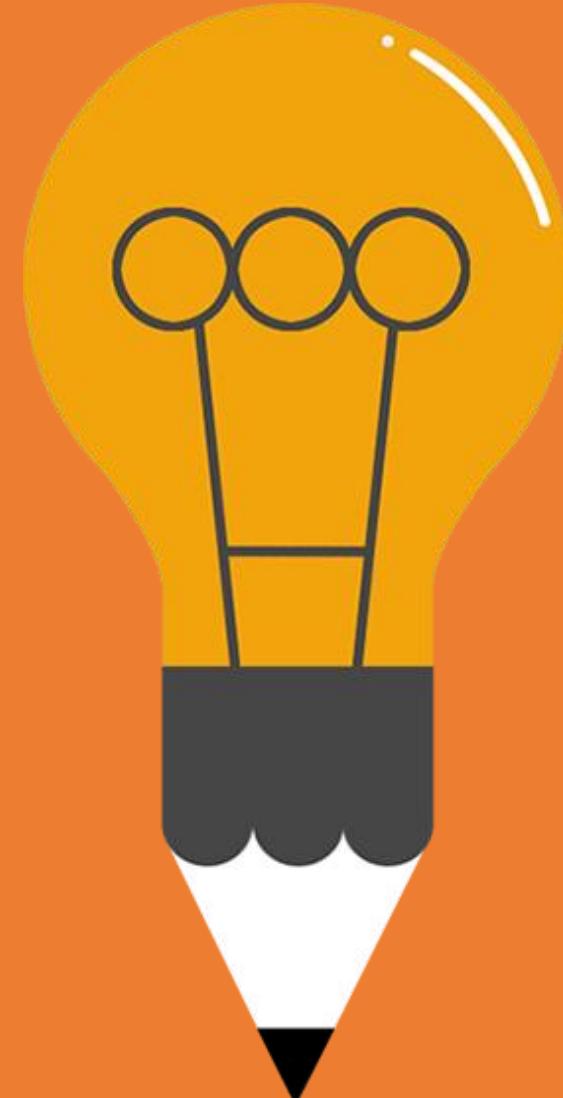
For each lecture:

- Presentation and video
- Accompanying Jupyter notebook
 - Explanation of the technique
 - Implementation in pandas and Numpy
 - Implementation in Scikit-learn (when possible)
 - Implementation in Feature-engine



Final Summary

- Final lecture comparing the performance of the different categorical encoding techniques with different machine learning models.
- Additional reading resources.





THANK YOU

www.trainindata.com





Train In Data

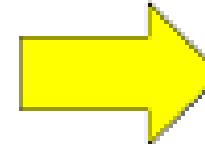
One hot encoding

One hot encoding: definition

- One hot encoding, consists in encoding each categorical variable with a set of boolean variables which take values 0 or 1, indicating if a category is present for each observation.

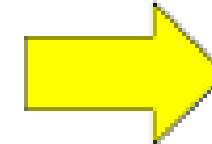


One hot encoding : example k dummy variables



Color	Red	Yellow	Green
Red	1	0	0
Red	1	0	0
Yellow	0	1	0
Green	0	0	1
Yellow			

One hot encoding : example k-1 dummy variables



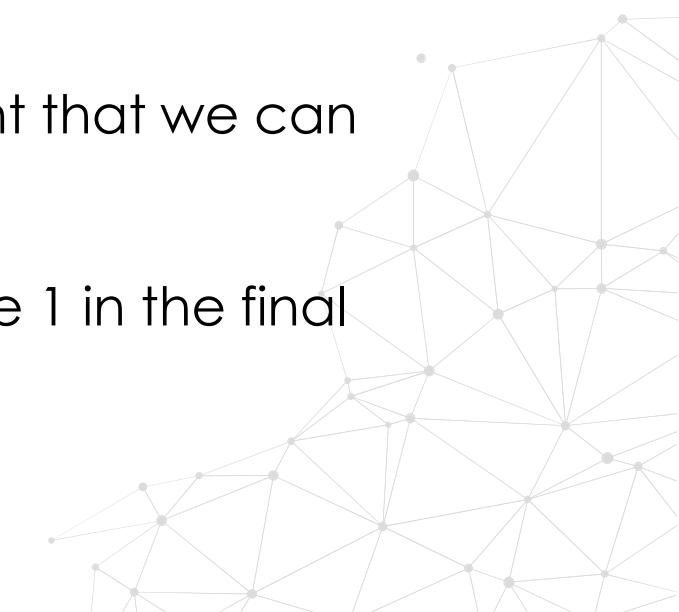
Color	Red	Yellow
Red	1	0
Red	1	0
Yellow	0	1
Green	0	0
Yellow	0	0

• One hot encoding into k - 1 variables

- More generally, a categorical variable should be encoded by creating $k-1$ binary variables, where k is the number of distinct categories.
- In the case of binary variables, like gender where $k=2$ (male / female) we need to create only 1 ($k - 1 = 1$) binary variable.

One hot encoding into $k-1$ binary variables takes into account that we can use 1 less dimension and still represent the whole information:

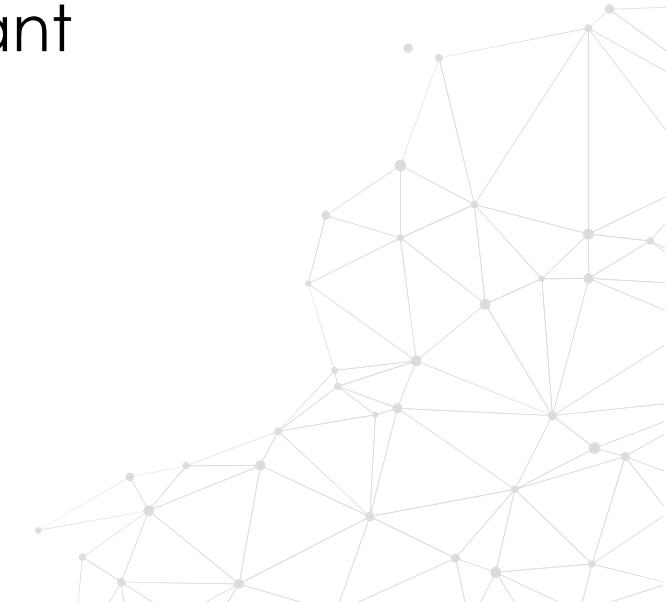
if the observation is 0 in all the binary variables, then it must be 1 in the final (not present) binary variable.



• One hot encoding into $k - 1$ variables

Most machine learning algorithms, consider **the entire data set** while being fit.

Therefore, encoding categorical variables into $k - 1$ binary variables, is better, as it avoids introducing redundant information.



One hot encoding into k variables

There are a few occasions when it is better to encode variables into k dummy variables:

- when building tree based algorithms
- when doing feature selection by recursive algorithms
- when interested in determine the importance of each single category



One hot encoding: Advantages

- Makes no assumption about the distribution or categories of the categorical variable
- Keeps all the information of the categorical variable
- Suitable for linear models



One hot encoding: Limitations

- Expands the feature space
- Does not add extra information while encoding
- Many dummy variables may be identical, introducing redundant information





THANK YOU

www.trainindata.com





One hot encoding of top categories

• One hot encoding of top categories: definition

- Performing one hot encoding, only considering the most frequent categories
- in the winning solution of the KDD 2009 cup: "[Winning the KDD Cup Orange Challenge with Ensemble Selection](#)", the authors limit one hot encoding to the 10 most frequent labels of the variable.



One hot encoding of top categories: example

Variable = City

London → 1000 observations

Manchester → 500 observations

Leeds → 200 Observations

Yorkshire, Milton-Keynes, Cambridge => 10 observations each



London	Manchester	Leeds
1	0	0
0	0	1
1	0	0
0	1	0
0	0	1
0	1	0
0	0	0

• One hot encoding of top categories: Advantages

- Straightforward to implement
- Does not require hrs of variable exploration
- Does not expand massively the feature space
- Handles new categories in test set
- Suitable for linear models



• One hot encoding of top categories: Limitations

- It does extend the feature space to some degree
- Does not add extra information while encoding
- Does not keep the information of the ignored labels





THANK YOU

www.trainindata.com





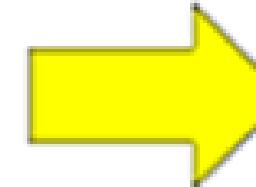
Label / Ordinal / Integer encoding

• Label / integers encoding: definition

- Integer encoding consist in replacing the categories by digits from 1 to n (or 0 to n-1, depending the implementation), where n is the number of distinct categories of the variable.
- The numbers are assigned arbitrarily.
- This encoding method allows for quick benchmarking of machine learning models.

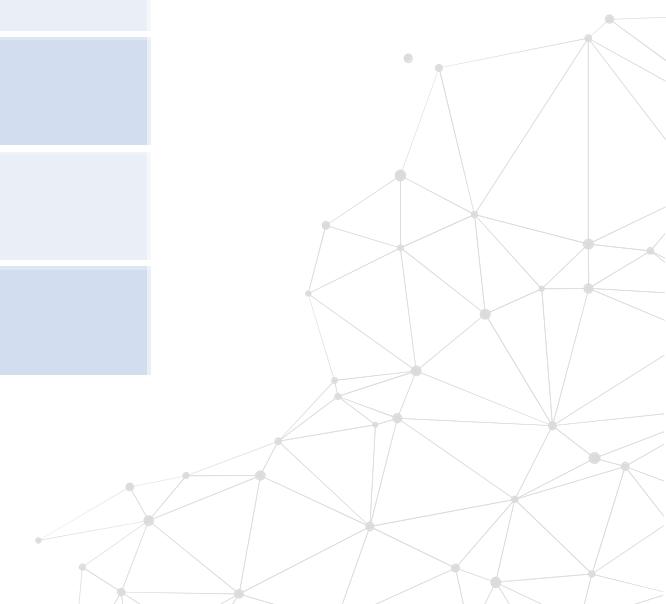


• Label / integers encoding : example



Color
Red
Red
Yellow
Green
Yellow

Color
1
1
2
3
2



• Label / integers encoding: Advantages

- Straightforward to implement
- Does not expand the feature space
- Can work well enough with tree based algorithms



• Label / integers encoding: Limitations

- Does not add extra information while encoding
- Not suitable for linear models
- Does not handle new categories in test set automatically





THANK YOU

www.trainindata.com





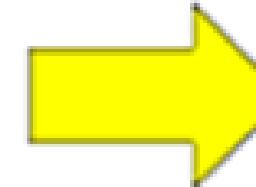
Label / Ordinal / Integer encoding

• Label / integers encoding: definition

- Integer encoding consist in replacing the categories by digits from 1 to n (or 0 to n-1, depending the implementation), where n is the number of distinct categories of the variable.
- The numbers are assigned arbitrarily.
- This encoding method allows for quick benchmarking of machine learning models.

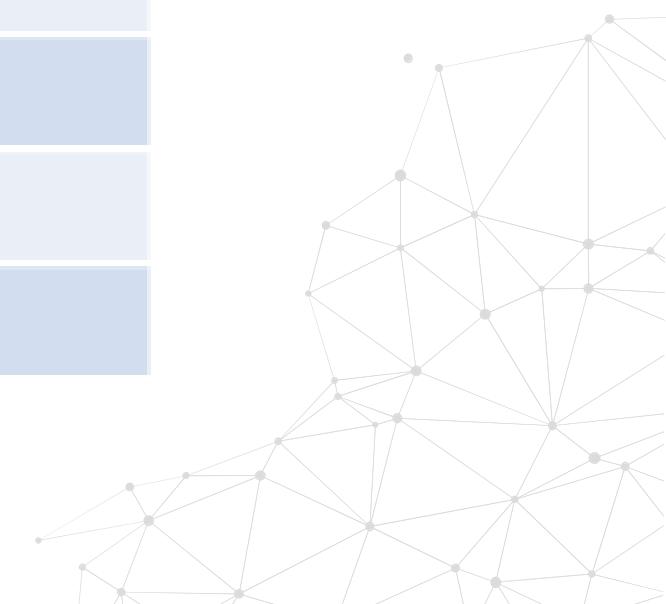


• Label / integers encoding : example



Color
Red
Red
Yellow
Green
Yellow

Color
1
1
2
3
2



• Label / integers encoding: Advantages

- Straightforward to implement
- Does not expand the feature space
- Can work well enough with tree based algorithms



• Label / integers encoding: Limitations

- Does not add extra information while encoding
- Not suitable for linear models
- Does not handle new categories in test set automatically





THANK YOU

www.trainindata.com





Count or frequency encoding

• Count / frequency encoding: definition

- Categories are replaced by the count or percentage of observations that show that category in the dataset.
- Captures the representation of each label in a dataset
- Very popular encoding method in Kaggle competitions.
- Assumption: the number observations shown by each category is predictive of the target.



Count encoding: example

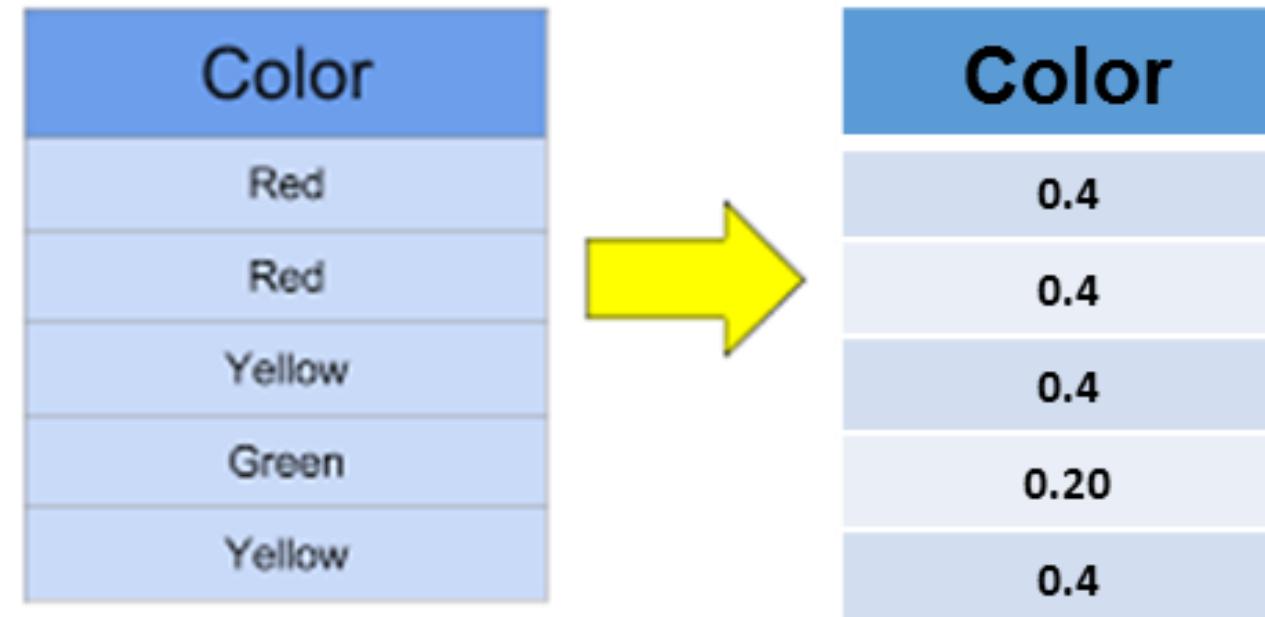


Color
Red
Red
Yellow
Green
Yellow

Color
2
2
2
1
2



Frequency encoding: example



Color
Red
Red
Yellow
Green
Yellow

→

Color
0.4
0.4
0.4
0.20
0.4



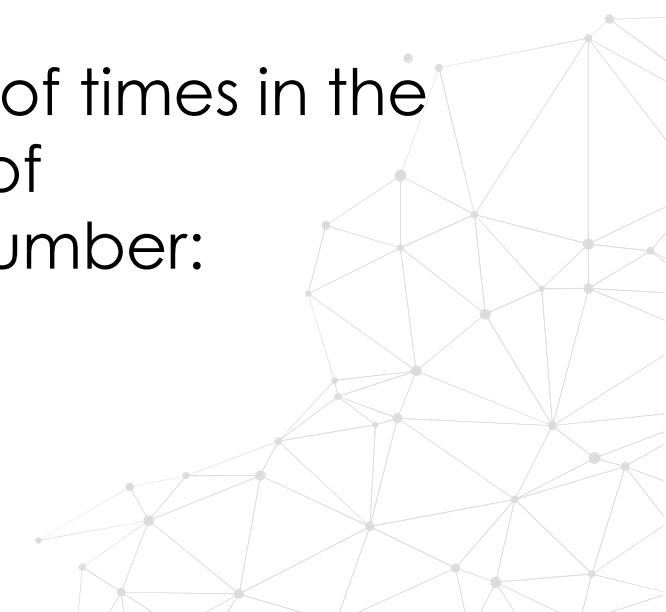
• Count / frequency encoding: Advantages

- Straightforward to implement
- Does not expand the feature space
- Can work well enough with tree based algorithms



• Count / frequency encoding: Limitations

- Not suitable for linear models
- Does not handle new categories in test set automatically
- If 2 different categories appear the same amount of times in the dataset, that is, they appear in the same number of observations, they will be replaced by the same number:
 - may lose valuable information.

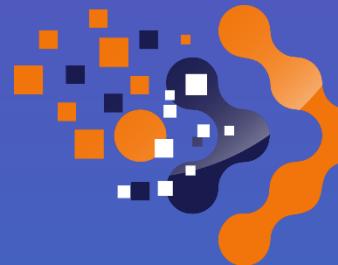




THANK YOU

www.trainindata.com





Train In Data

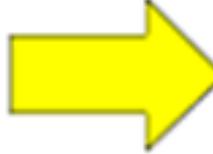
Ordered ordinal encoding

• Ordered ordinal encoding: definition

- Categories are replaced by integers from 1 to k, where k is the number of distinct categories in the variable, but this numbering is informed by the mean of the target for each category.



Ordered ordinal encoding: example



Color	Target	Color
Red	0	2
Red	1	2
Yellow	1	1
Green	0	3
Yellow	1	1

• Ordered ordinal encoding: Advantages

- Straightforward to implement
- Does not expand the feature space
- Creates monotonic relationship between categories and target



Ordered ordinal encoding: Limitations

- May lead to over-fitting





THANK YOU

www.trainindata.com





Train In Data

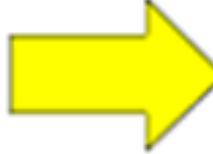
Ordered ordinal encoding

• Ordered ordinal encoding: definition

- Categories are replaced by integers from 1 to k, where k is the number of distinct categories in the variable, but this numbering is informed by the mean of the target for each category.



Ordered ordinal encoding: example



Color	Target	Color
Red	0	2
Red	1	2
Yellow	1	1
Green	0	3
Yellow	1	1

• Ordered ordinal encoding: Advantages

- Straightforward to implement
- Does not expand the feature space
- Creates monotonic relationship between categories and target



Ordered ordinal encoding: Limitations

- May lead to over-fitting





THANK YOU

www.trainindata.com





Train In Data

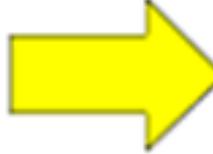
Ordered ordinal encoding

• Ordered ordinal encoding: definition

- Categories are replaced by integers from 1 to k, where k is the number of distinct categories in the variable, but this numbering is informed by the mean of the target for each category.



Ordered ordinal encoding: example



Color	Target	Color
Red	0	2
Red	1	2
Yellow	1	1
Green	0	3
Yellow	1	1

• Ordered ordinal encoding: Advantages

- Straightforward to implement
- Does not expand the feature space
- Creates monotonic relationship between categories and target



Ordered ordinal encoding: Limitations

- May lead to over-fitting





THANK YOU

www.trainindata.com





Train In Data

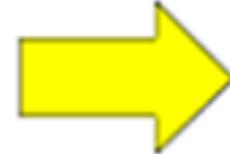
Mean or target encoding

• Mean encoding: definition

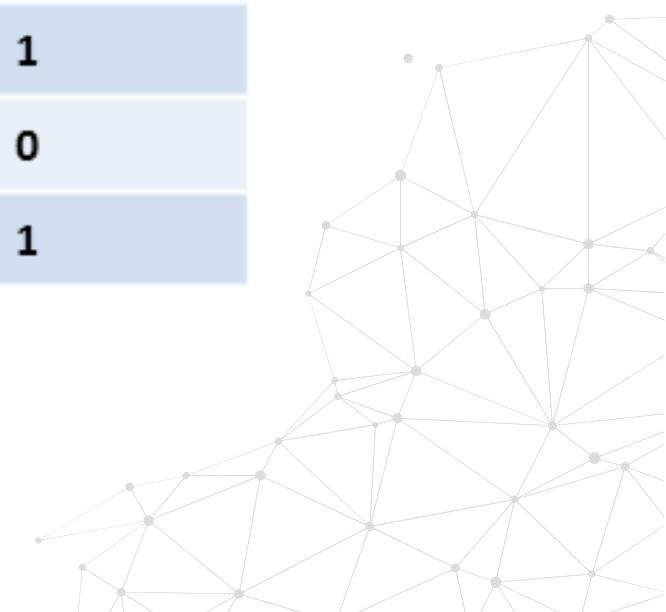
- Mean encoding implies replacing the category by the average target value for that category



• Mean encoding: example



Color	Target	Color
Red	0	0.5
Red	1	0.5
Yellow	1	1
Green	0	0
Yellow	1	1



• Mean encoding: Advantages

- Straightforward to implement
- Does not expand the feature space
- Creates monotonic relationship between categories and target



• Mean encoding: Limitations

- May lead to over-fitting
- If 2 categories show the same mean of target, they will be replaced by the same number => potential loss of value



• Mean encoding with Category Encoders

Category Encoders

latest

Search docs

Backward Difference Coding

BaseN

Binary

CatBoost Encoder

Hashing

Helmert Coding

James-Stein Encoder

Leave One Out

M-estimate

One Hot

Ordinal

Polynomial Coding

Sum Coding

Docs » Target Encoder

[View page source](#)

Target Encoder

```
class category_encoders.target_encoder.TargetEncoder(verbose=0, cols=None, drop_invariant=False, return_df=True, handle_missing='value', handle_unknown='value', min_samples_leaf=1, smoothing=1.0)
    [source] 🔍
```

Target encoding for categorical features.

For the case of categorical target: features are replaced with a blend of posterior probability of the target given particular categorical value and the prior probability of the target over all the training data.

For the case of continuous target: features are replaced with a blend of the expected value of the target given particular categorical value and the expected value of the target over all the training data.

Parameters

verbose: int

integer indicating verbosity of the output. 0 for none.



THANK YOU

www.trainindata.com





Train In Data

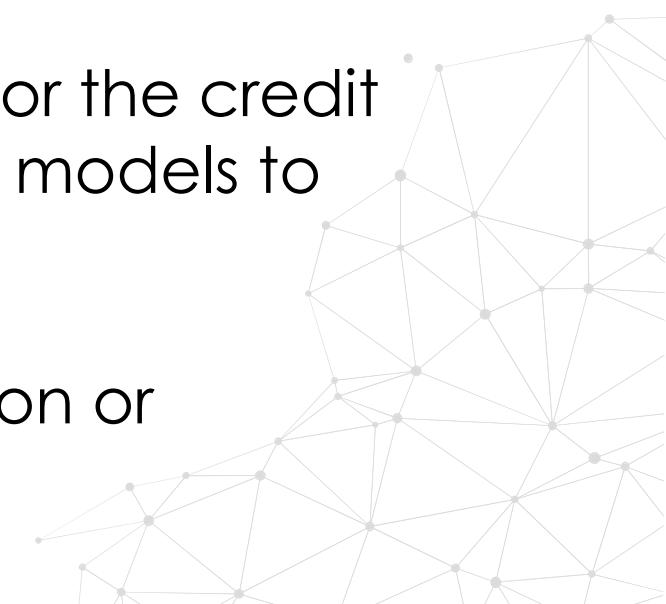
Weight of Evidence

• Weight of evidence: definition

$$WoE = \ln\left(\frac{\text{Proportion of good events}}{\text{Proportion of bad events}}\right)$$

Weight of Evidence (WoE) was developed primarily for the credit and financial industries to help build more predictive models to evaluate the risk of loan default.

That is, to predict how likely the money lent to a person or institution is to be lost.



• Weight of evidence: definition

$$WoE = \ln\left(\frac{\text{Proportion of good events}}{\text{Proportion of bad events}}\right)$$

- Proportion of good events:
 - sum of + observations per category / total positive observations
- Proportion of bad events:
 - sum of - observations per category / total negative observations



• Weight of evidence: example

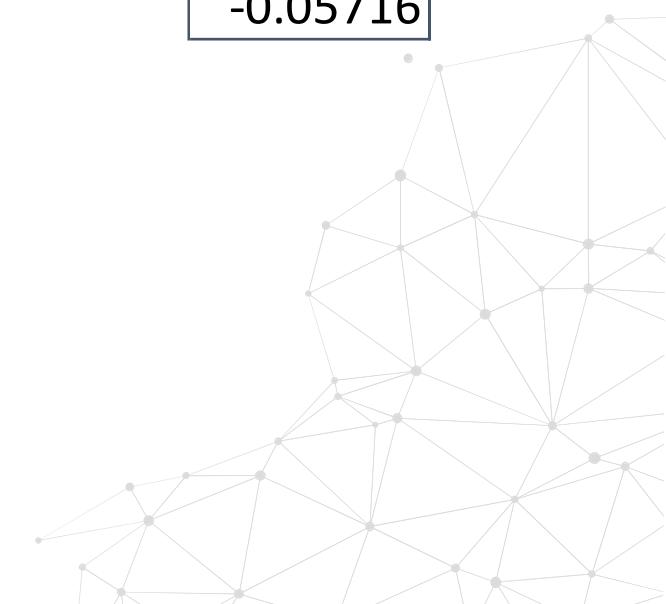
	survived	non-survived
A	50	30
B	75	40
C	25	15
total	150	85
total passengers		235



% good	% bad
0.33	0.35
0.50	0.47
0.17	0.18



WoE
-0.05716
0.060625
-0.05716



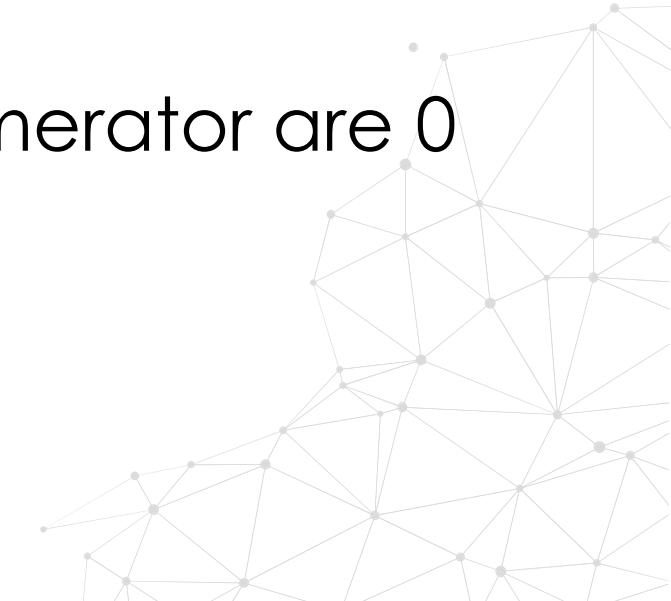
• Weight of evidence: Advantages

- Creates a monotonic relationship between the target and the variables.
- It orders the categories on a "logistic" scale which is natural for logistic regression
- The transformed variables can then be compared because they are on the same scale.
 - Therefore, it is possible to determine which one is more predictive.



• Weight of evidence: Limitations

- May lead to over-fitting
- Not defined when the denominator or numerator are 0



• Weight of evidence: open source

The screenshot shows the documentation for the `category_encoders.woe.WOEEncoder` class. The page title is "Weight of Evidence". It includes a code snippet for the class definition:

```
class category_encoders.woe.WOEEncoder(verbose=0, cols=None, drop_invariant=False, return_df=True, handle_unknown='value', handle_missing='value', random_state=None, randomized=False, sigma=0.05, regularization=1.0) [source]
```

Below the code, there's a brief description: "Weight of Evidence coding for categorical features." The "Parameters" section lists four parameters with their descriptions:

- `verbose: int`: integer indicating verbosity of the output. 0 for none.
- `cols: list`: a list of columns to encode, if None, all string columns will be encoded.
- `drop_invariant: bool`: boolean for whether or not to drop columns with 0 variance.
- `return_df: bool`

The screenshot shows the documentation for the `WoERatioCategoricalEncoder` class. The page title is "WoERatioCategoricalEncoder". It includes a code snippet for a script named `titanic.py`:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

from feature_engine import categorical_encoders as ce

# Load dataset
def load_titanic():
    data = pd.read_csv('https://www.openml.org/data/get_csv/16826755/phpYEkMl')
    data = data.replace('?', np.nan)
    data['cabin'] = data['cabin'].astype(str).str[0]
    data['pclass'] = data['pclass'].astype('O')
    data['embarked'].fillna('C', inplace=True)
    return data

data = load_titanic()

# Separate into train and test sets
```

Docs » Categorical variable encoding » WoERatioCategoricalEncoder [Edit on GitHub](#)

WoERatioCategoricalEncoder

The `WoERatioCategoricalEncoder()` replaces the labels by the weight of evidence or the ratio of probabilities. It only works for binary classification.

The weight of evidence is given by: $\text{np.log}(p(1) / p(0))$

The target probability ratio is given by: $p(1) / p(0)$

The `CountFrequencyCategoricalEncoder()` works only with categorical variables. A list of variables can be indicated, or the encoder will automatically select all categorical variables in the train set.



THANK YOU

www.trainindata.com

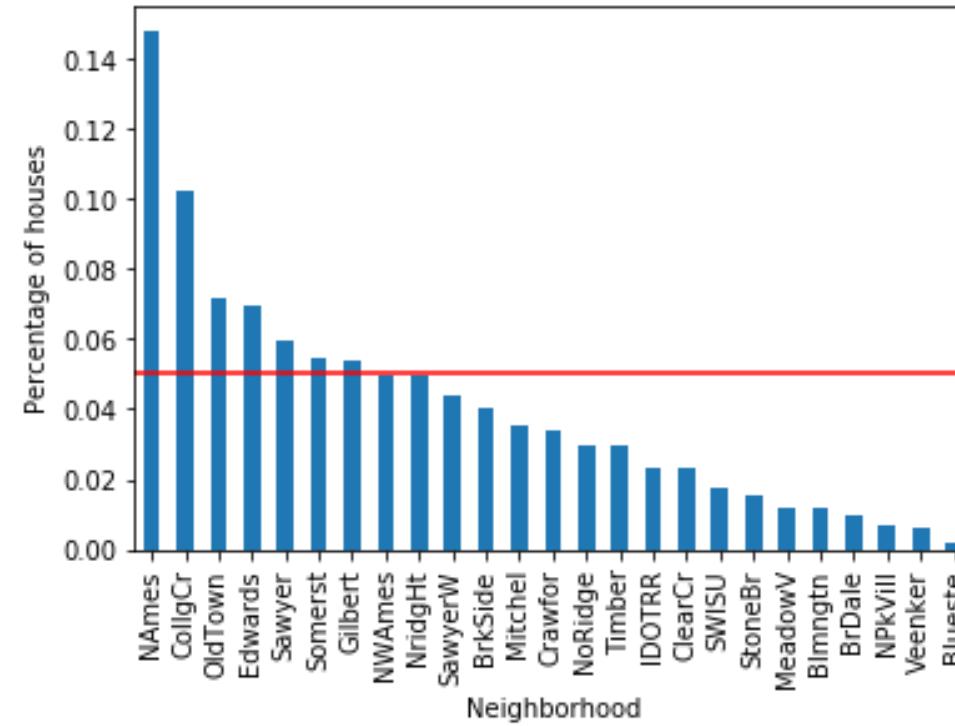




Rare Label Encoding

Grouping Rare Labels

- Rare labels are those that appear only in a tiny proportion of the observations in a dataset



Grouping Rare Labels

- Rare labels are those that appear only in a tiny proportion of the observations in a dataset



Some Scenarios

- Variables with one predominant category
- Variables with few categories
- Variables with high cardinality

Street	
Grvl	0.004892
Pave	0.995108

ExterQual	
Ex	0.029354
Fa	0.011742
Gd	0.332681
TA	0.626223

	Exterior2nd
AsbShng	0.016634
AsphShn	0.000978
Brk Cmn	0.003914
BrkFace	0.017613
CBlock	0.000978
CmentBd	0.038160
HdBoard	0.137965
ImStucc	0.007828
MetalSd	0.133072
Other	0.000978
Plywood	0.109589
Stone	0.003914
Stucco	0.015656
VinylSd	0.345401
Wd Sdng	0.138943
Wd Shng	0.028376

Some Scenarios

- Variables with one predominant category

Street	
Grvl	0.004892
Pave	0.995108



Street	
Rare	0.004892
Pave	0.995108



Some Scenarios

- Variables with few categories
 - Worth corroborating value of rare labels

ExterQual
Ex 0.029354
Fa 0.011742
Gd 0.332681
TA 0.626223

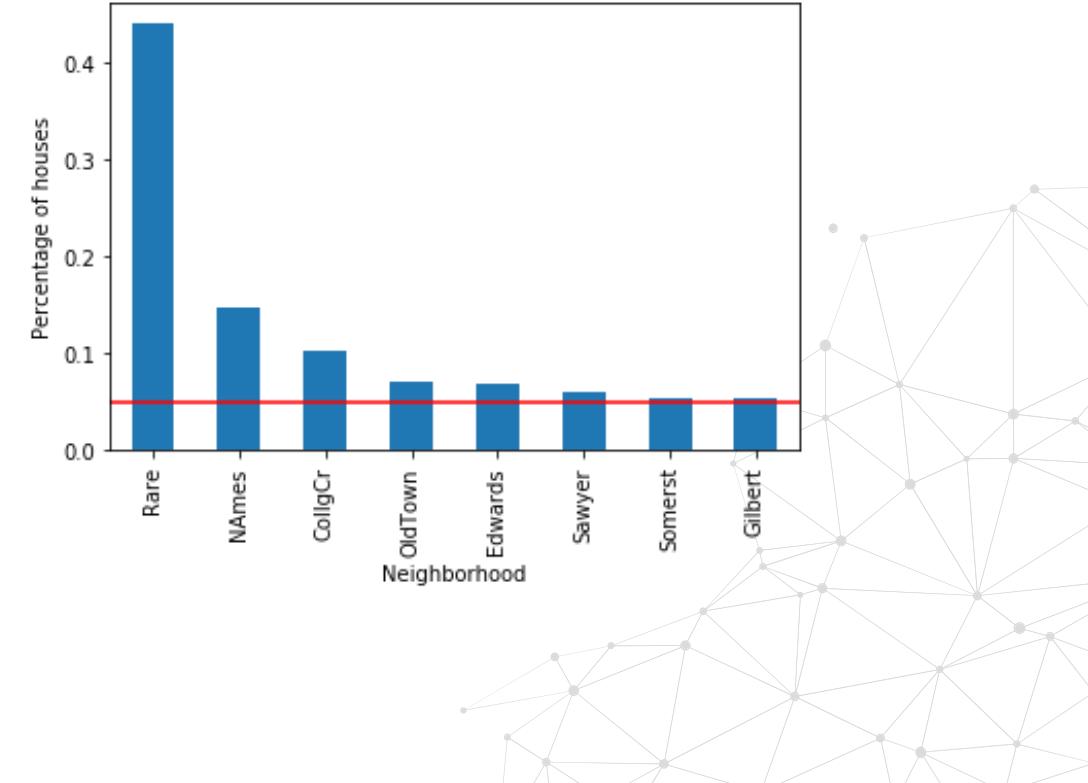
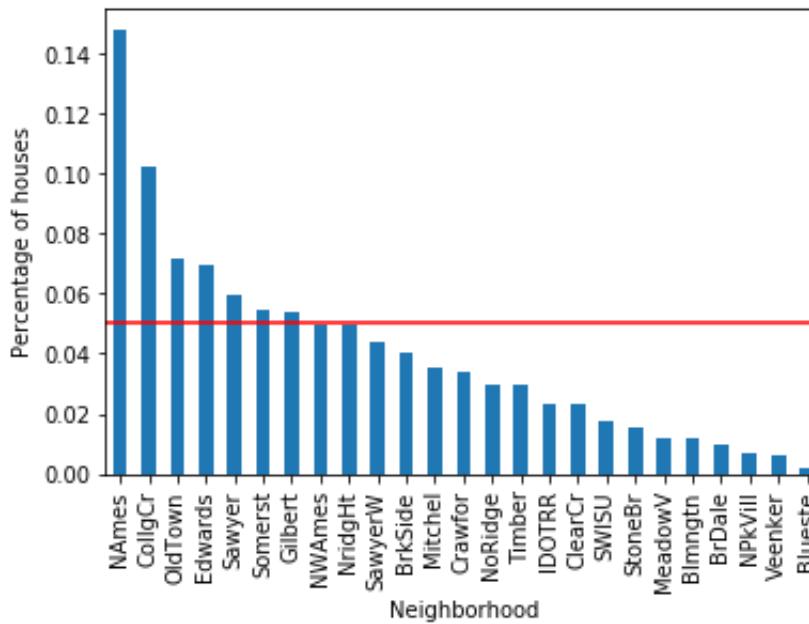


ExterQual
Rare 0.04
Gd 0.332681
TA 0.626223



Some Scenarios

- Variables with high cardinality



• Code should capture frequent categories

- This way, categories that are new in test set, will be treated as rare and put into the Rare group
- Model will know how to handle those categories as well, even though they were not present in the train set





THANK YOU

www.trainindata.com

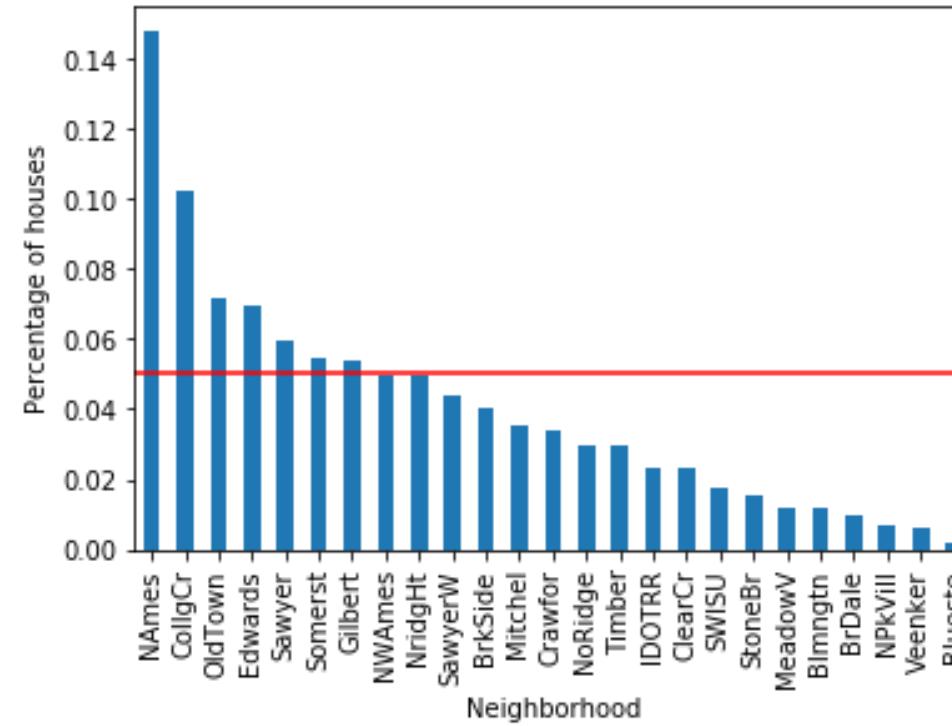




Rare Label Encoding

Grouping Rare Labels

- Rare labels are those that appear only in a tiny proportion of the observations in a dataset



Grouping Rare Labels

- Rare labels are those that appear only in a tiny proportion of the observations in a dataset



Some Scenarios

- Variables with one predominant category
- Variables with few categories
- Variables with high cardinality

Street	
Grvl	0.004892
Pave	0.995108

ExterQual	
Ex	0.029354
Fa	0.011742
Gd	0.332681
TA	0.626223

	Exterior2nd
AsbShng	0.016634
AsphShn	0.000978
Brk Cmn	0.003914
BrkFace	0.017613
CBlock	0.000978
CmentBd	0.038160
HdBoard	0.137965
ImStucc	0.007828
MetalSd	0.133072
Other	0.000978
Plywood	0.109589
Stone	0.003914
Stucco	0.015656
VinylSd	0.345401
Wd Sdng	0.138943
Wd Shng	0.028376

Some Scenarios

- Variables with one predominant category

Street	
Grvl	0.004892
Pave	0.995108



Street	
Rare	0.004892
Pave	0.995108



Some Scenarios

- Variables with few categories
 - Worth corroborating value of rare labels

ExterQual
Ex 0.029354
Fa 0.011742
Gd 0.332681
TA 0.626223

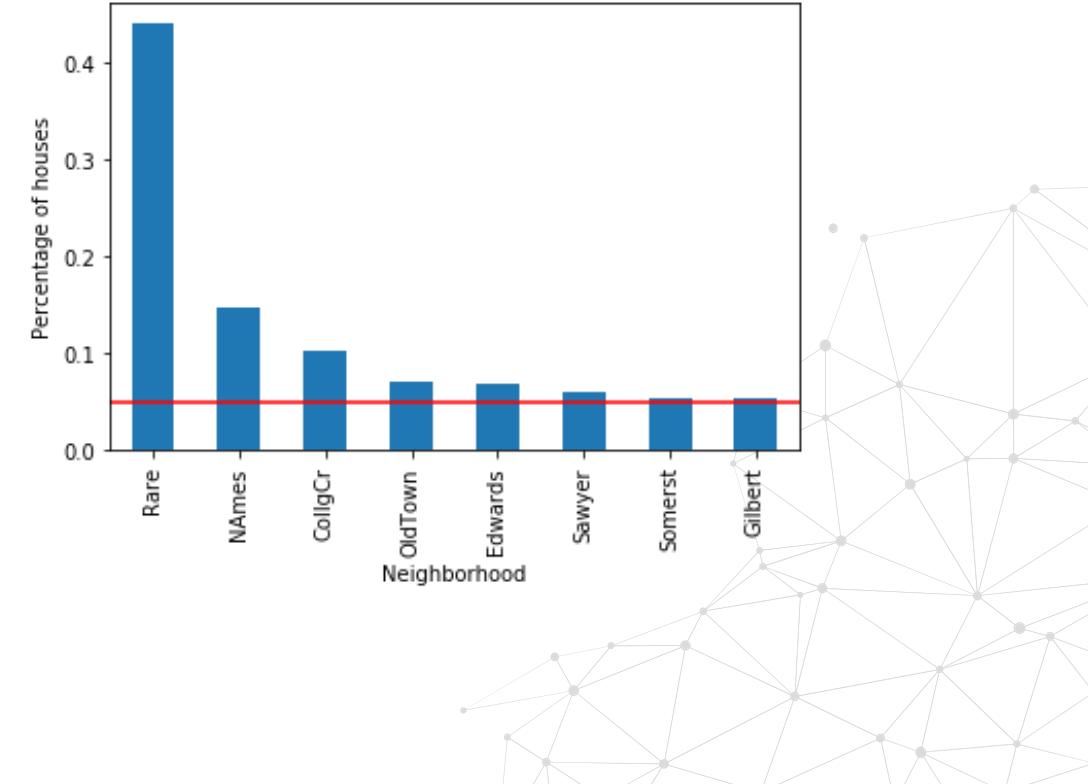
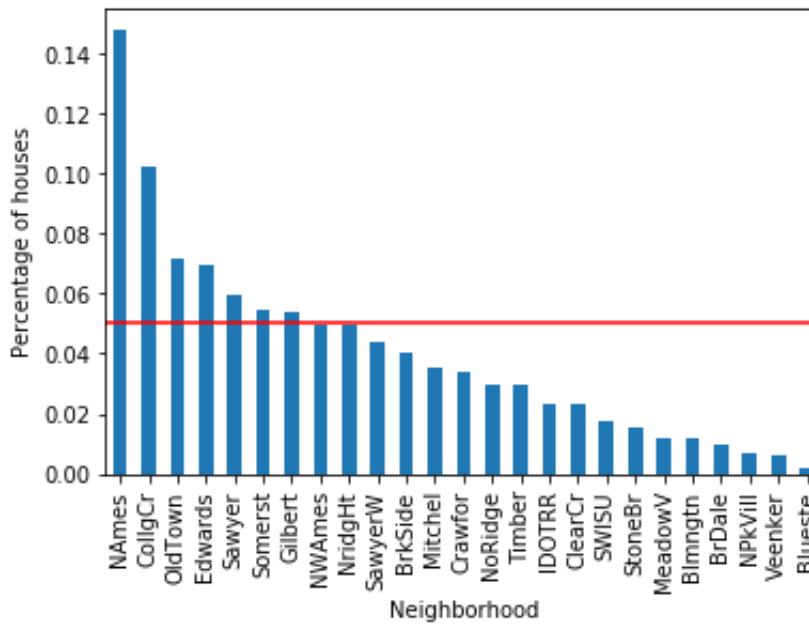


ExterQual
Rare 0.04
Gd 0.332681
TA 0.626223



Some Scenarios

- Variables with high cardinality



• Code should capture frequent categories

- This way, categories that are new in test set, will be treated as rare and put into the Rare group
- Model will know how to handle those categories as well, even though they were not present in the train set





THANK YOU

www.trainindata.com

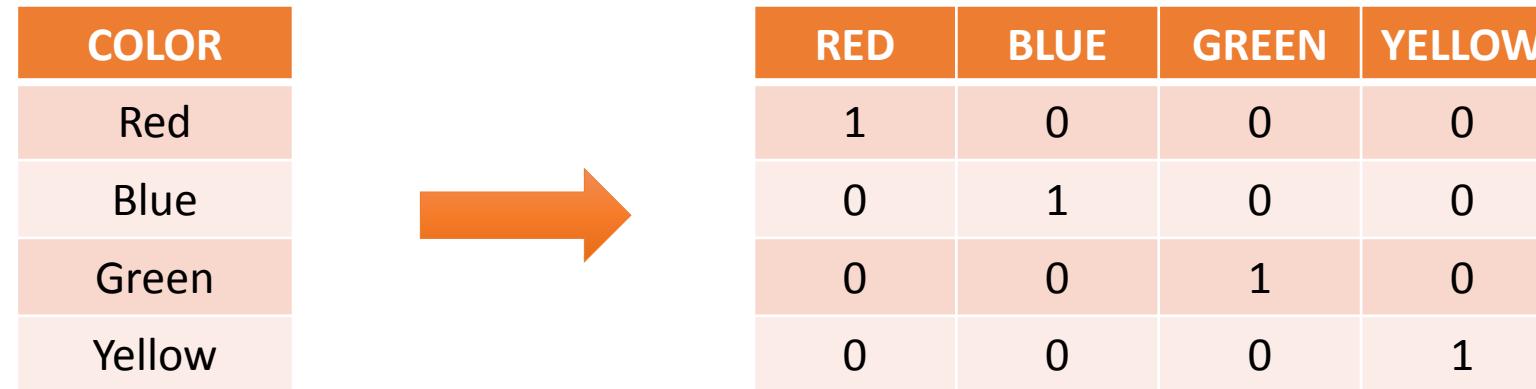




Binary encoding and Feature Hashing

One hot encoding: definition

- Create 1 binary variable per category.
- Each single derived variable has a meaning on its own.



COLOR	RED	BLUE	GREEN	YELLOW
Red	1	0	0	0
Blue	0	1	0	0
Green	0	0	1	0
Yellow	0	0	0	1



• Binary encoding: definition

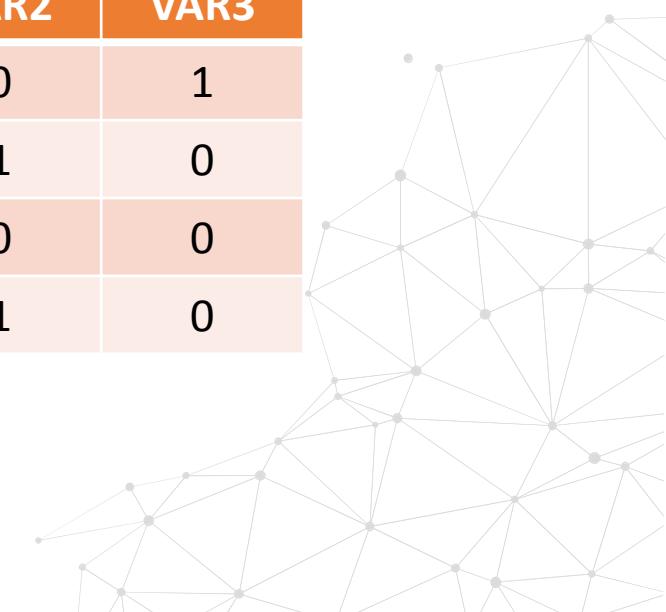
- Use binary code, a collection of 0s and 1s, to encode the meaning of the variable.
- Individually derived variable lack human readable meaning.



• Feature hashing: definition

- Use a hashing method, any of choice, to encode the variables
- May lead to different labels taking the same value

COLOR	HASH	VAR1	VAR2	VAR3
Red	2	0	0	1
Blue	1	0	1	0
Green	0	1	0	0
Yellow	1	0	1	0



Binary encoding: Category encoders

← → ⌂ 🔒 contrib.scikit-learn.org/categorical-encoding/binary.html

Category Encoders latest Search docs

Backward Difference Coding BaseN Binary CatBoost Encoder Hashing Helmert Coding James-Stein Encoder Leave One Out M-estimate One Hot Ordinal Polynomial Coding Sum Coding Target Encoder

Docs » Binary View page source

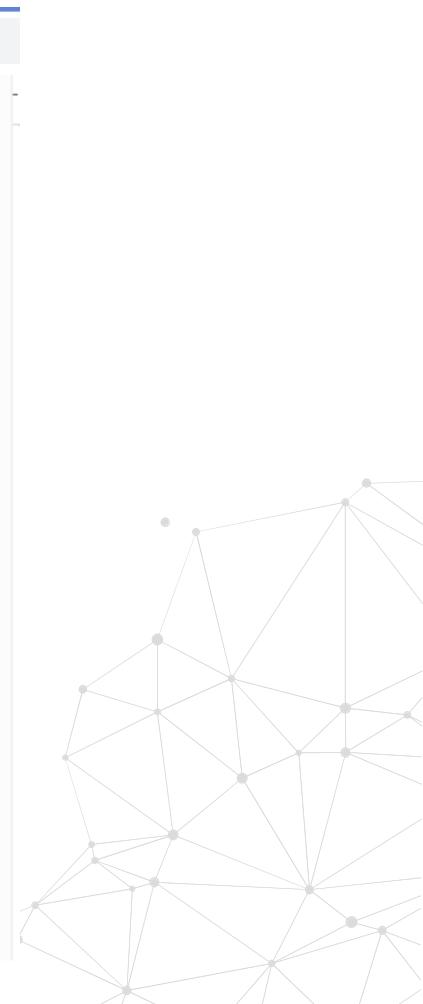
Binary

```
class category_encoders.binary.BinaryEncoder(verbose=0, cols=None, mapping=None, drop_invariant=False, return_df=True, handle_unknown='value', handle_missing='value') [source]
```

Binary encoding for categorical variables, similar to onehot, but stores categories as binary bitstrings.

Parameters

- verbose: int**
integer indicating verbosity of the output. 0 for none.
- cols: list**
a list of columns to encode, if None, all string columns will be encoded.
- drop_invariant: bool**
boolean for whether or not to drop columns with 0 variance.
- return_df: bool**



Feature hashing: Category encoders

← → ⌂ contrib.scikit-learn.org/categorical-encoding/hashing.html

Category Encoders

latest

Search docs

- Backward Difference Coding
- BaseN
- Binary
- CatBoost Encoder

Hashing

- Helmert Coding
- James-Stein Encoder
- Leave One Out
- M-estimate
- One Hot
- Ordinal
- Polynomial Coding
- Sum Coding
- Target Encoder

Docs » Hashing

[View page source](#)

Hashing

```
class category_encoders.hashing.HashingEncoder(verbose=0, n_components=8, cols=None, drop_invariant=False, return_df=True, hash_method='md5') [source]
```

A basic multivariate hashing implementation with configurable dimensionality/precision.

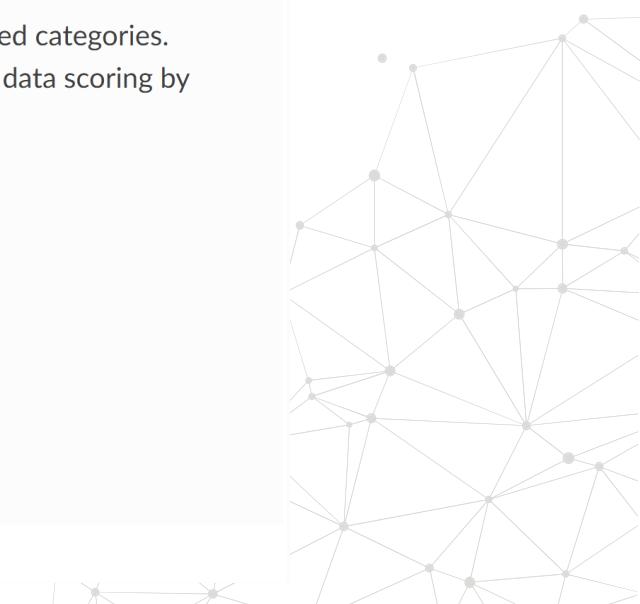
The advantage of this encoder is that it does not maintain a dictionary of observed categories. Consequently, the encoder does not grow in size and accepts new values during data scoring by design.

Parameters

verbose: int
integer indicating verbosity of the output. 0 for none.

cols: list
a list of columns to encode, if None, all string columns will be encoded.

drop_invariant: bool





THANK YOU

www.trainindata.com

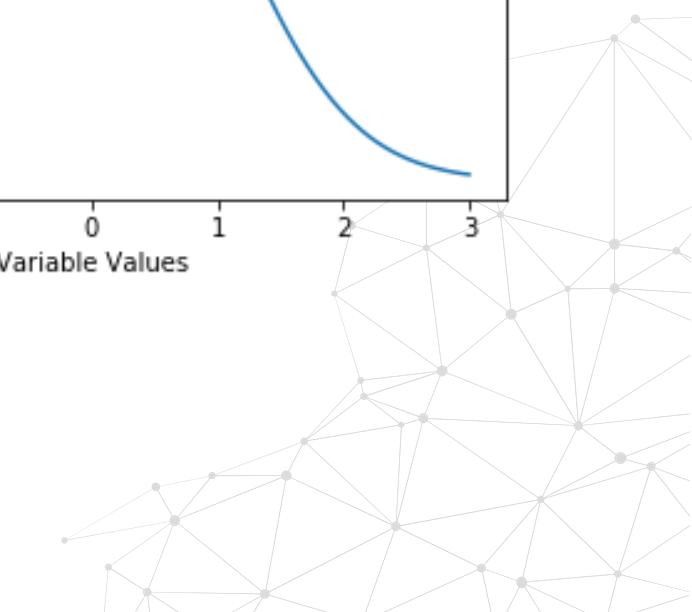
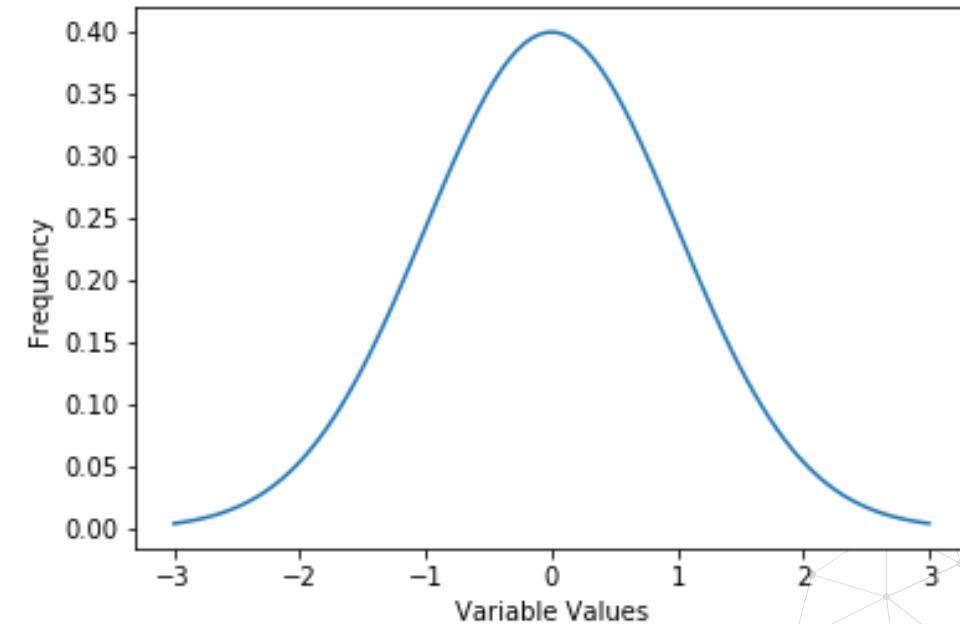




Variable Transformation

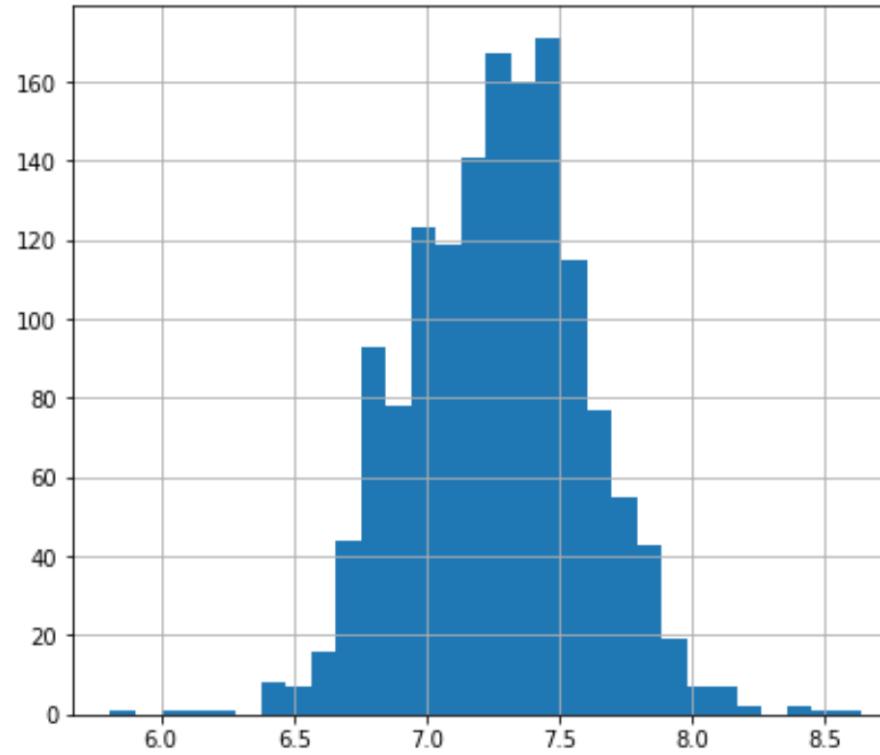
Normality in linear models

- Variables follow a Gaussian Distribution
- Normality can be assessed with histograms and Q-Q plots

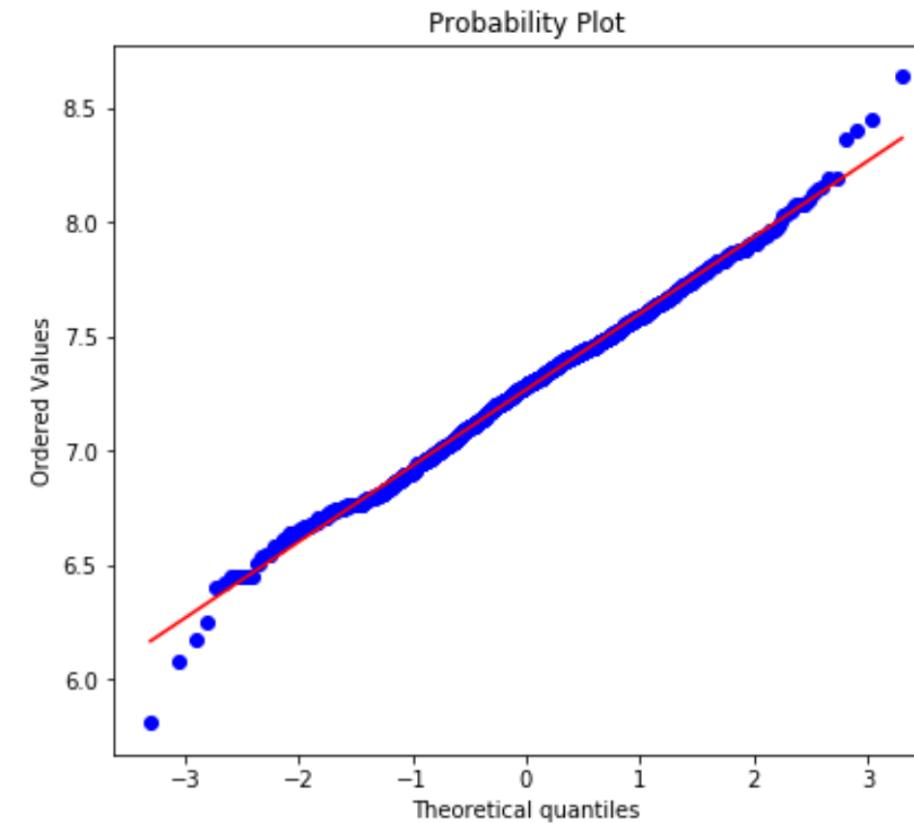


Normality assessment

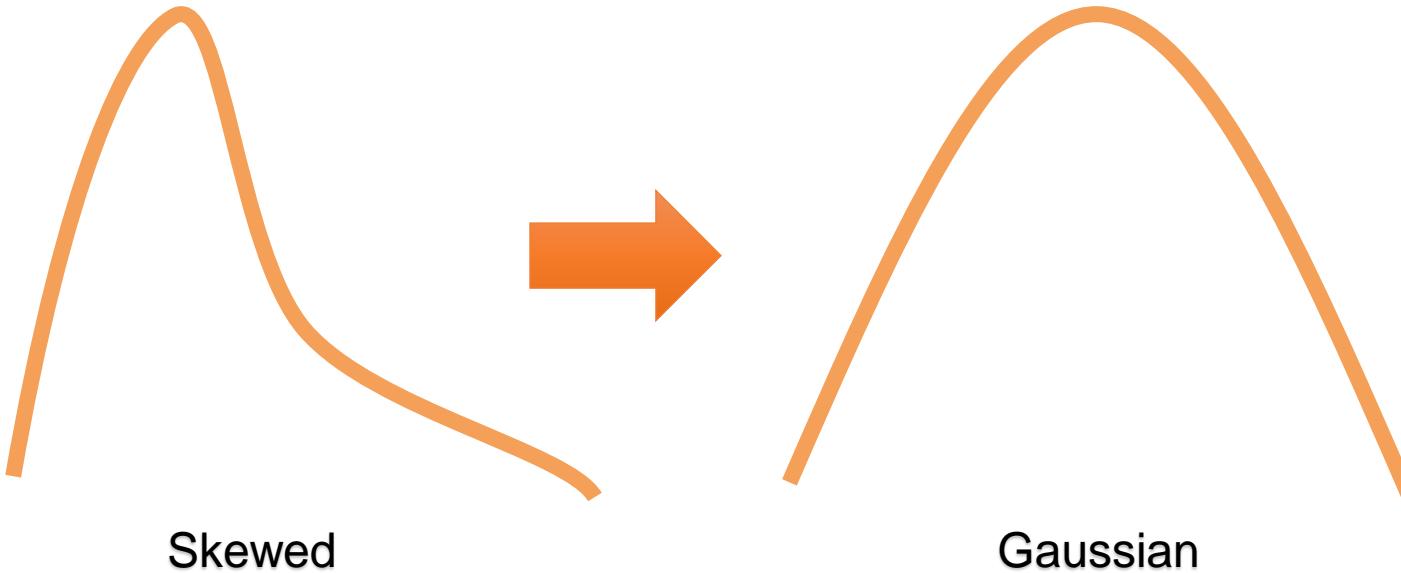
Histogram



Q-Q plot

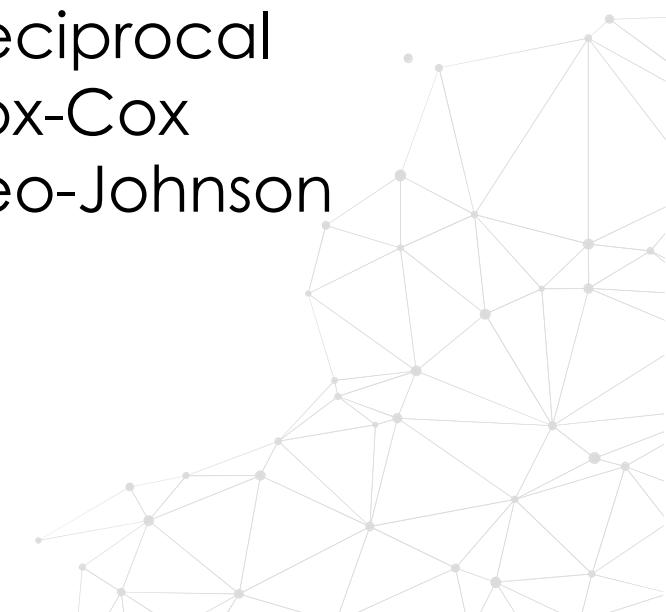


Mathematical transformations



Variable transformation

- Logarithmic
- Exponential
- Reciprocal
- Box-Cox
- Yeo-Johnson



Mathematical transformations

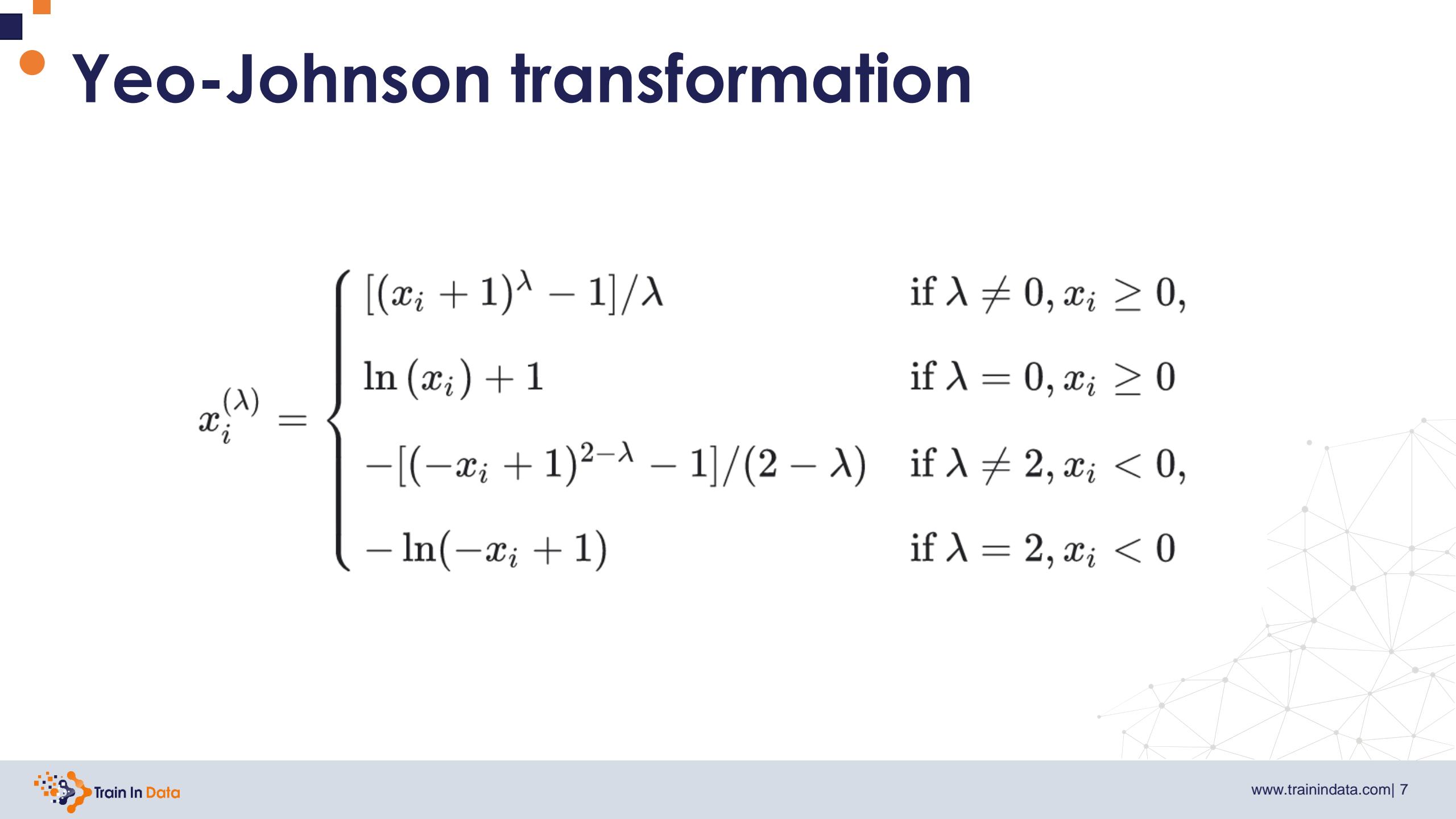
Logarithmic	Reciprocal	Power / Exponential	Exponential special cases
<input type="checkbox"/> Log(X)	<input type="checkbox"/> $1 / X$	<input type="checkbox"/> $X \exp(\lambda)$	<input type="checkbox"/> Box-Cox ($X > 0$)
<input type="checkbox"/> $X > 0$	<input type="checkbox"/> $X \neq 0$	<input type="checkbox"/> $\text{sqr}(X) / \text{cube}(X)$	<input type="checkbox"/> Yeo-Johnson
		<input type="checkbox"/> Not defined for all X	



• Box-Cox transformation

$$x_i^{(\lambda)} = \begin{cases} \frac{x_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln(x_i) & \text{if } \lambda = 0, \end{cases}$$





• Yeo-Johnson transformation

$$x_i^{(\lambda)} = \begin{cases} [(x_i + 1)^\lambda - 1]/\lambda & \text{if } \lambda \neq 0, x_i \geq 0, \\ \ln(x_i) + 1 & \text{if } \lambda = 0, x_i \geq 0 \\ -[(-x_i + 1)^{2-\lambda} - 1]/(2 - \lambda) & \text{if } \lambda \neq 2, x_i < 0, \\ -\ln(-x_i + 1) & \text{if } \lambda = 2, x_i < 0 \end{cases}$$



THANK YOU

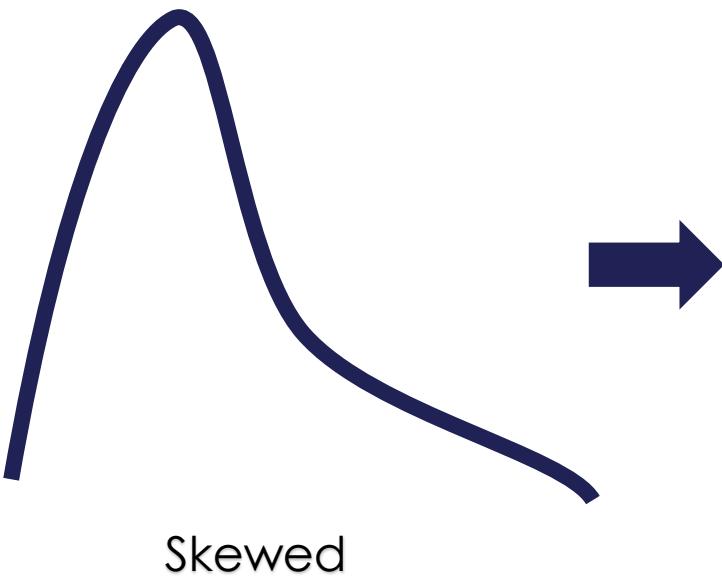
www.trainindata.com





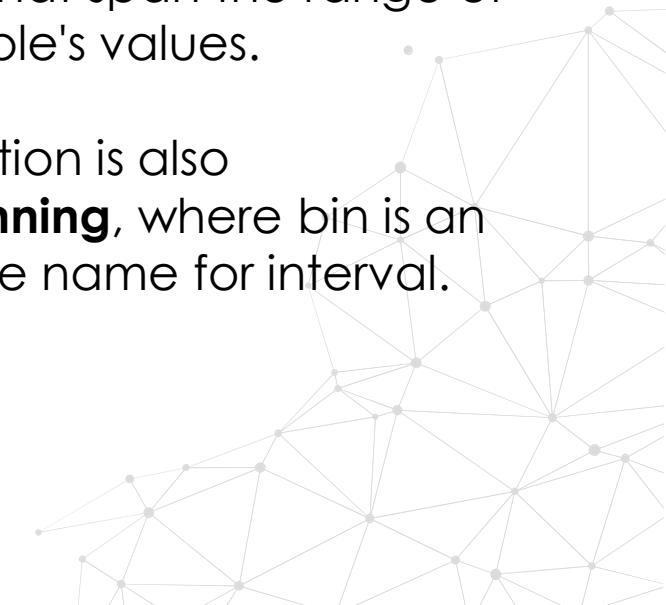
Discretisation

• Discretisation

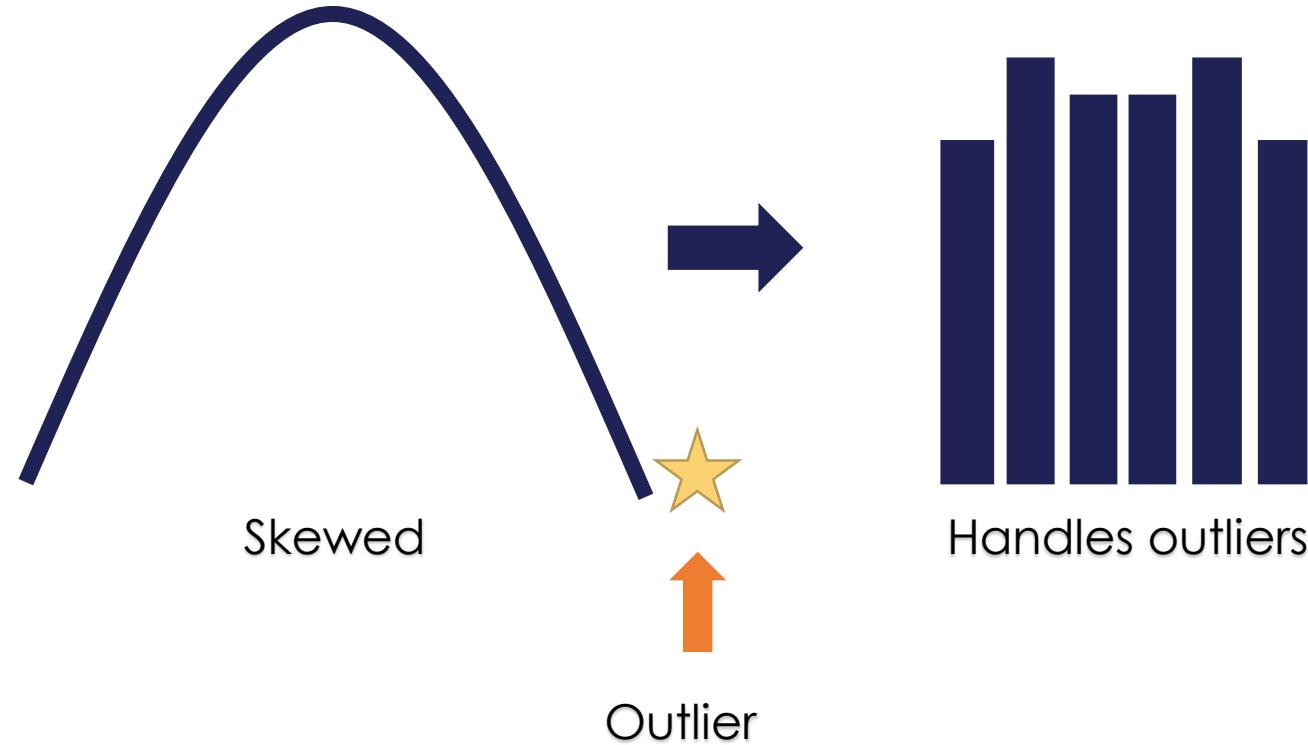


Discretisation is the process of transforming continuous variables into discrete variables by creating a set of contiguous intervals that span the range of the variable's values.

Discretisation is also called **binning**, where bin is an alternative name for interval.



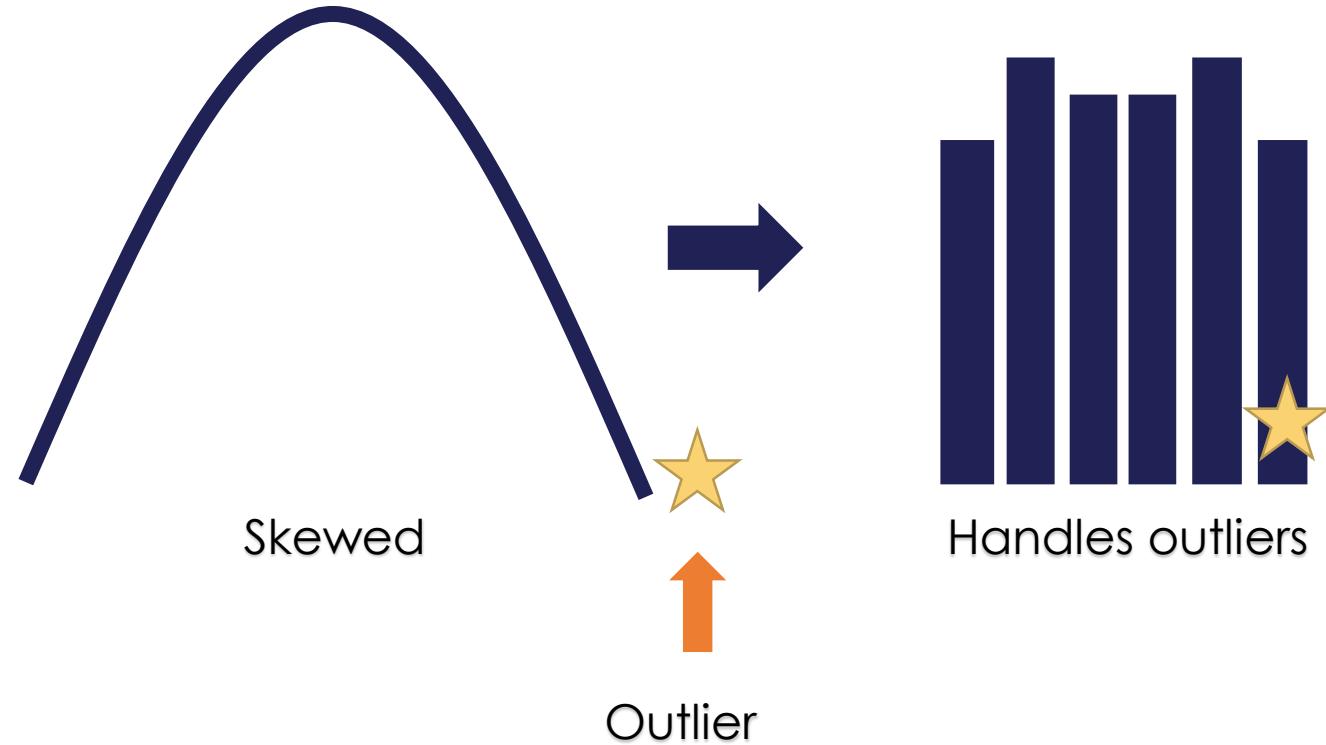
• Discretisation



Outliers are placed into the lower or upper intervals, together with the remaining inlier values at the ends of the distribution.



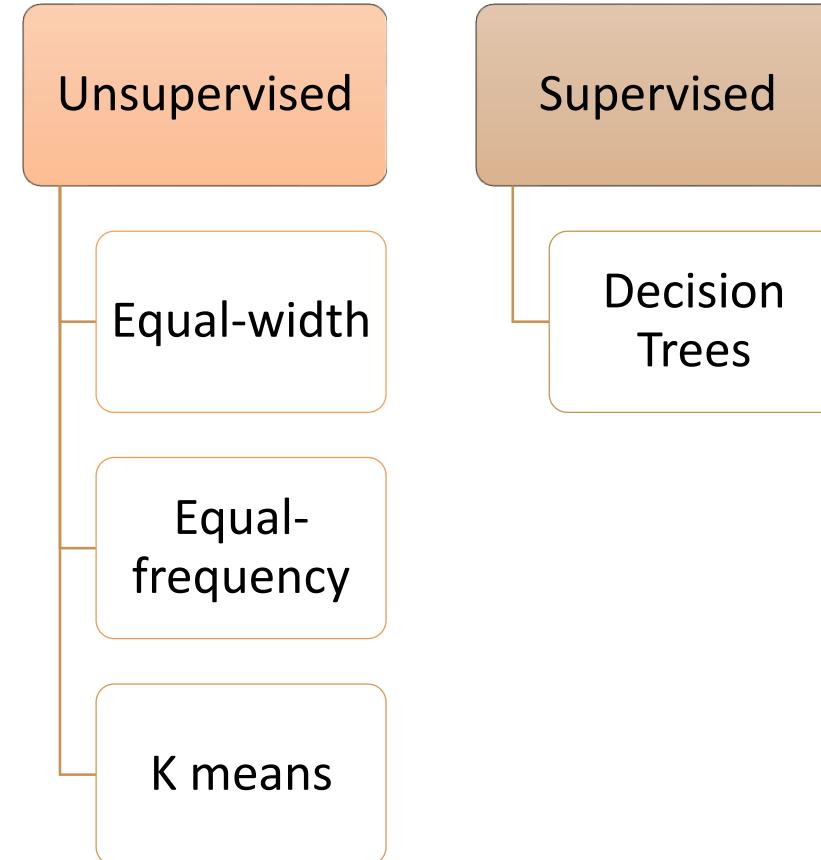
• Discretisation



Outliers are placed into the lower or upper intervals, together with the remaining inlier values at the ends of the distribution.



Discretisation approaches



• Accompanying Jupyter Notebook



- Accompanying Jupyter Notebook
- How to perform discretisation:
 - Pandas
 - Scikit-learn
 - Feature-engine
- Effect of discretisation on variable distributions





THANK YOU

www.trainindata.com





Equal-width discretisation

• Equal-width discretisation: definition

Equal width discretisation divides the scope of possible values into N bins of the same width.

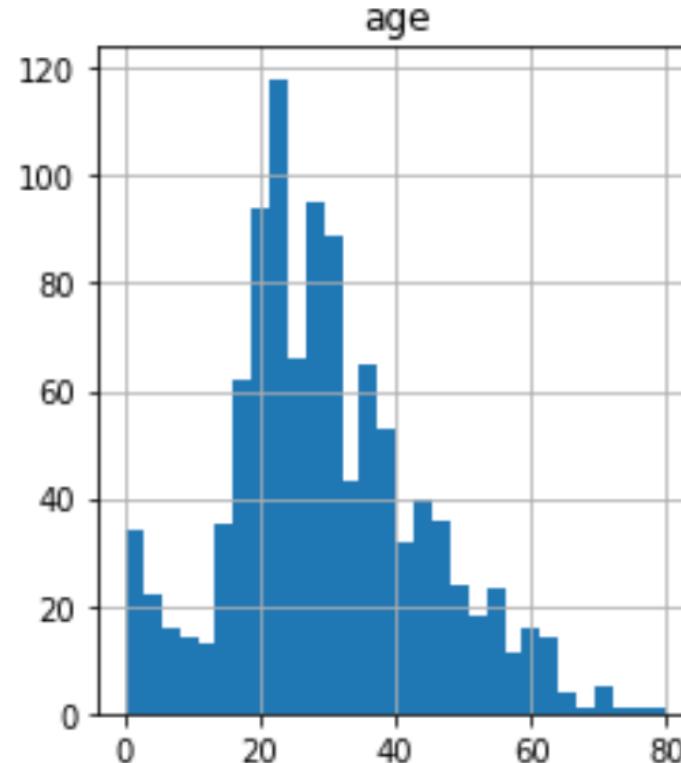
The interval width is determined by:

$$\text{width} = (\text{max value} - \text{min value}) / N$$

where N is the number of bins or intervals.



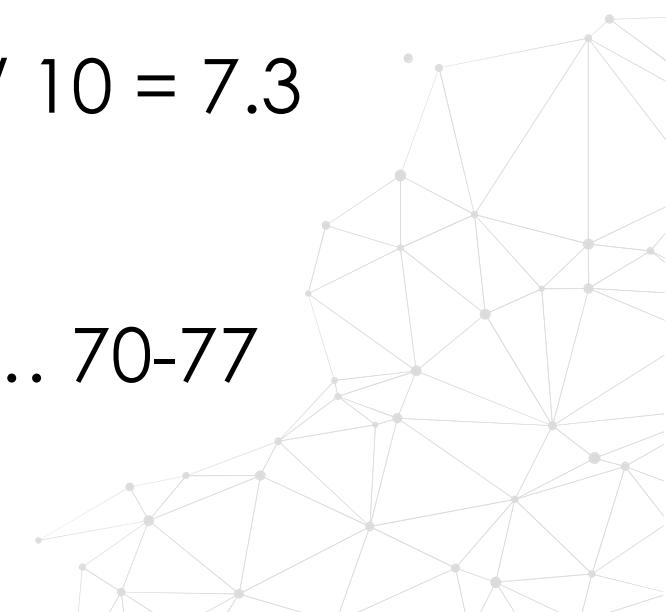
• Equal-width discretisation: calculation



Min age = 0
Max age = 73
Intervals = 10

$$\text{width} = (73 - 0) / 10 = 7.3$$

Intervals:
0-7; 7-14; 14-21 ... 70-77



• Equal-width discretisation: calculation

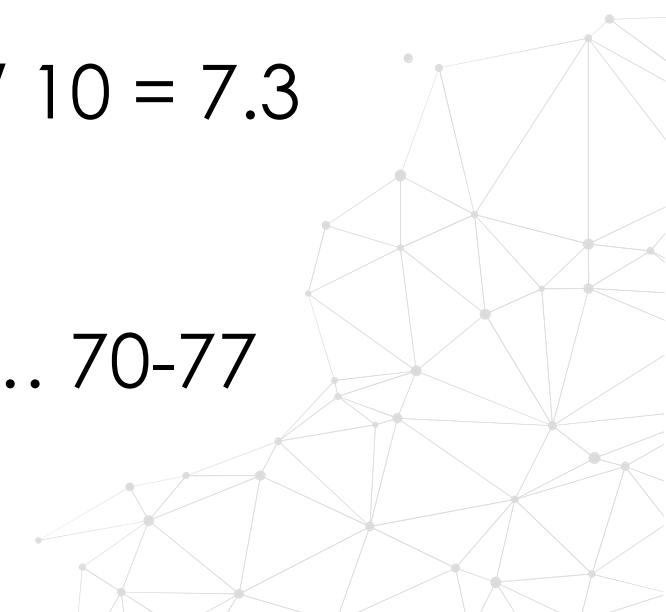
age	Age_disc
13.0	(7.0, 14.0]
4.0	(-0.001, 7.0]
30.0	(28.0, 35.0]
21.0	(14.0, 21.0]
22.0	(21.0, 28.0]
16.0	(14.0, 21.0]
30.0	(28.0, 35.0]
2.0	(-0.001, 7.0]
49.0	(42.0, 49.0]
35.0	(28.0, 35.0]



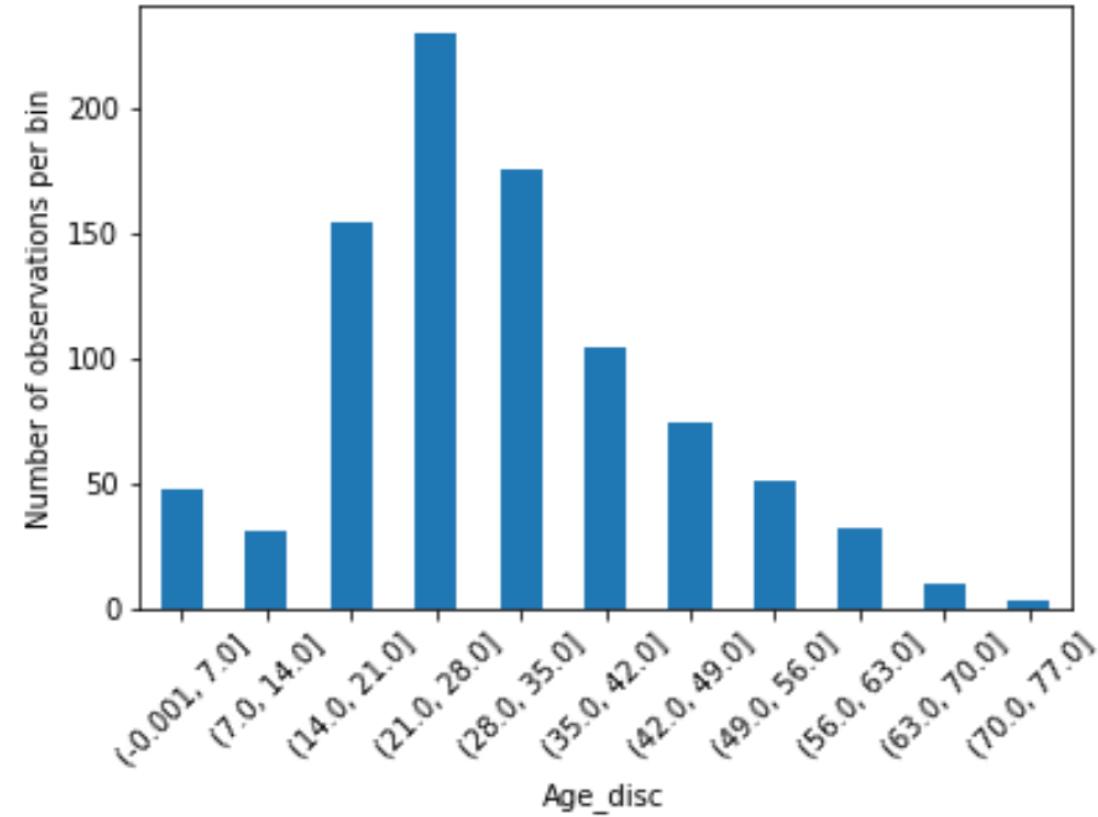
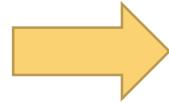
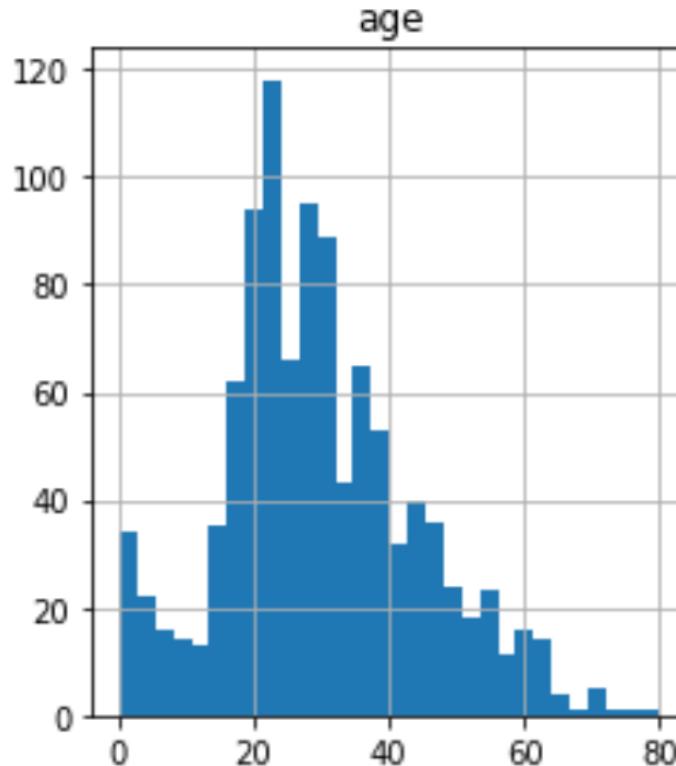
Min age = 0
Max age = 73
Intervals = 10

$$\text{width} = (73 - 0) / 10 = 7.3$$

Intervals:
0-7; 7-14; 14-21 ... 70-77



• Equal-width discretisation: calculation



• Equal-width discretisation: summary

- Does not improve value spread
- Handles outliers
- Creates discrete variable
- Good to combine with categorical encodings





THANK YOU

www.trainindata.com





Train In Data

Equal-frequency discretisation

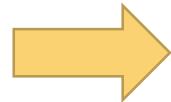
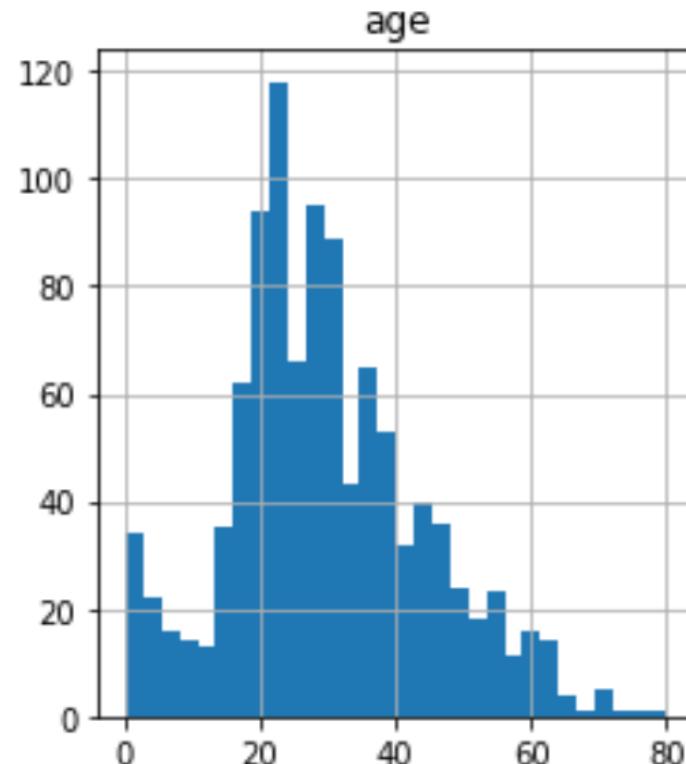
• Equal-frequency discretisation: definition

Equal frequency discretisation divides the scope of possible values of the variable into N bins, where each bin carries the same amount of observations.

Interval boundaries correspond to the quantiles.



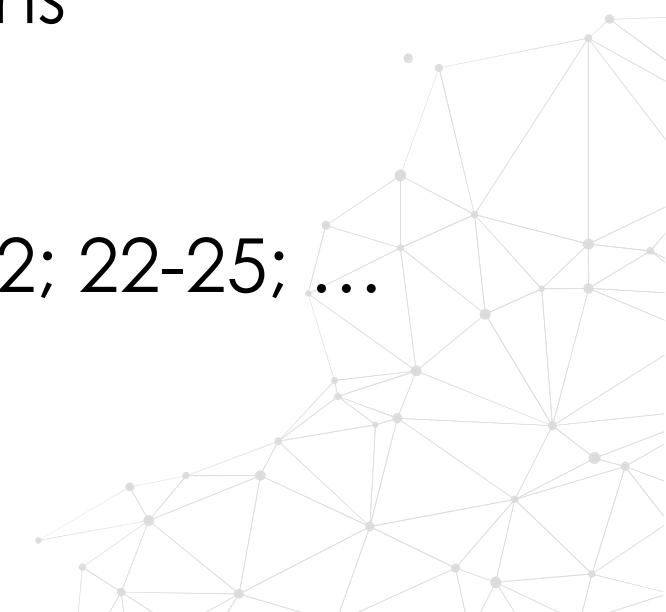
• Equal-frequency discretisation: definition



Intervals = 10

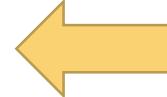
Each interval contains 10% of total observations

Intervals:
0-16; 16-20; 20-22; 22-25; ...
50-74



• Equal-frequency discretisation: definition

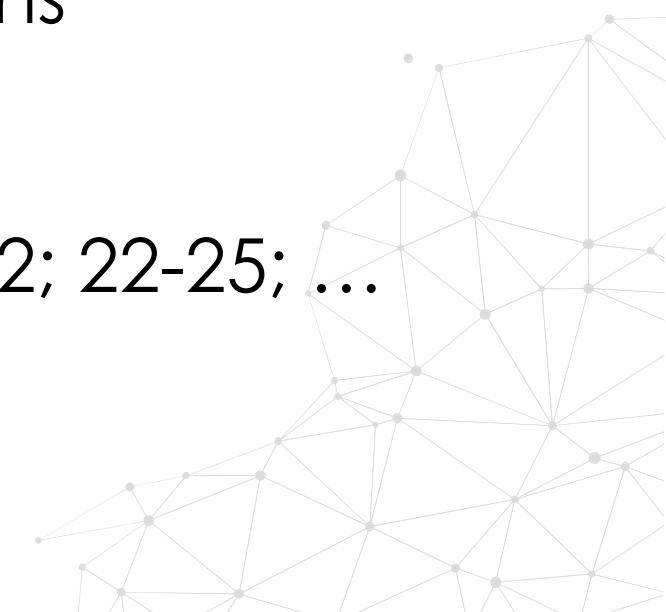
age	Age_disc
38.0	(36.0, 42.0]
21.0	(20.0, 22.25]
42.0	(36.0, 42.0]
34.0	(31.0, 36.0]
25.0	(22.25, 25.0]
4.0	(0.167, 16.0]
48.0	(42.0, 50.0]
52.0	(50.0, 74.0]
57.0	(50.0, 74.0]
32.0	(31.0, 36.0]



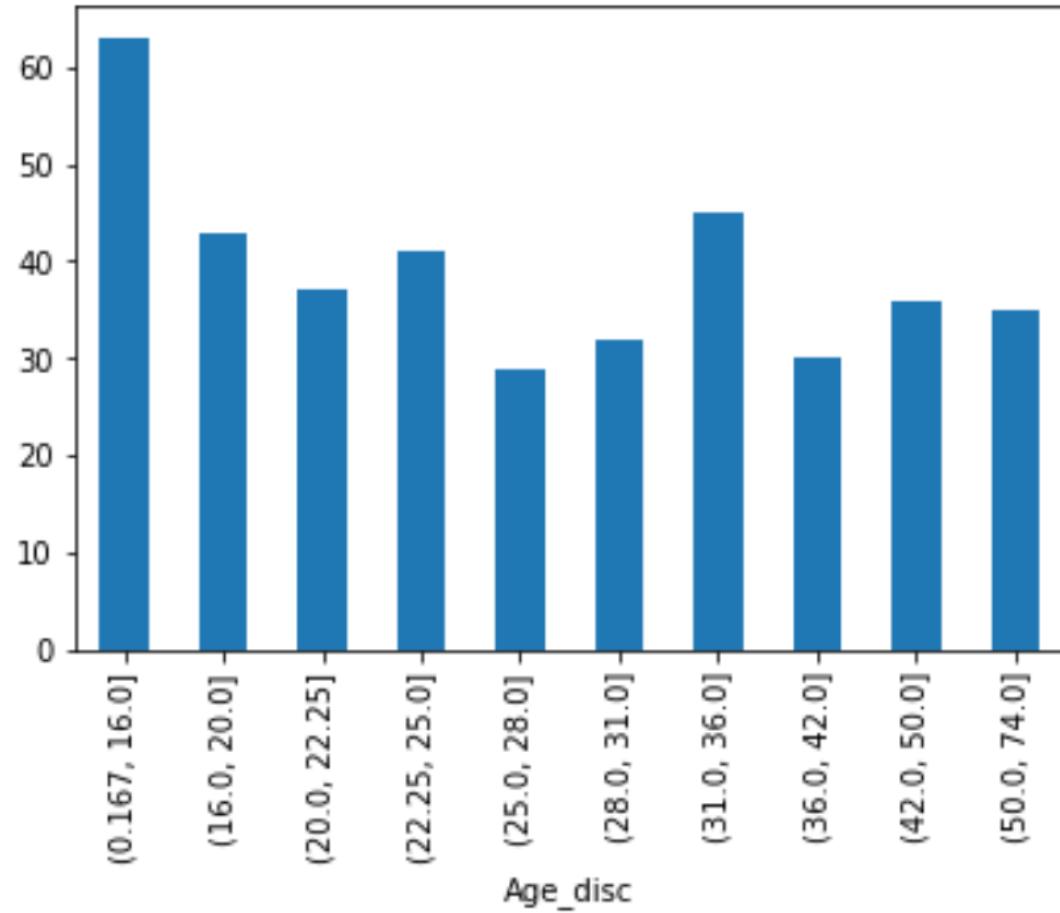
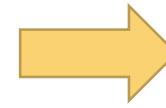
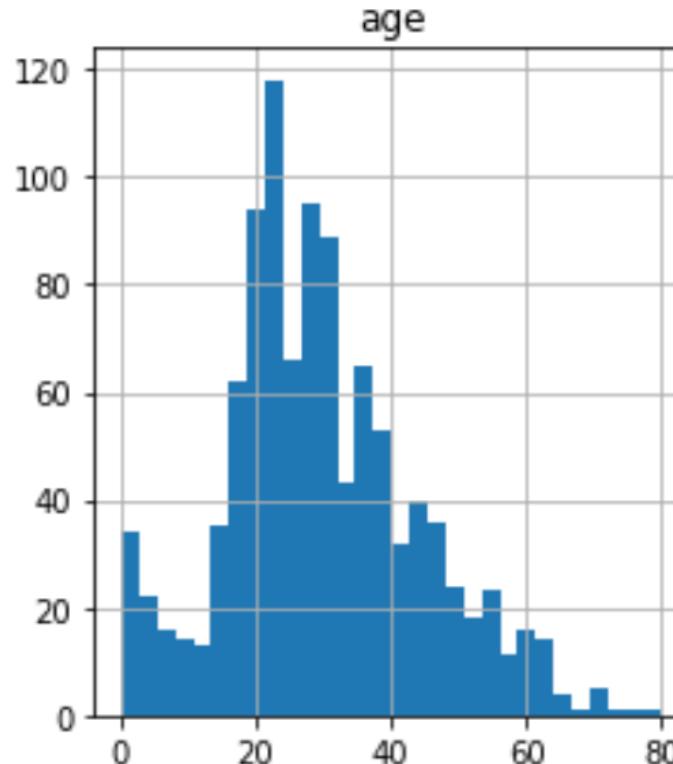
Intervals = 10

Each interval contains 10% of total observations

Intervals:
0-16; 16-20; 20-22; 22-25; ...
50-74



• Equal-frequency discretisation: definition



• Equal-frequency discretisation: summary

- Improve value spread
- Handles outliers
- Creates discrete variable
- Good to combine with categorical encodings





THANK YOU

www.trainindata.com





K-means discretisation

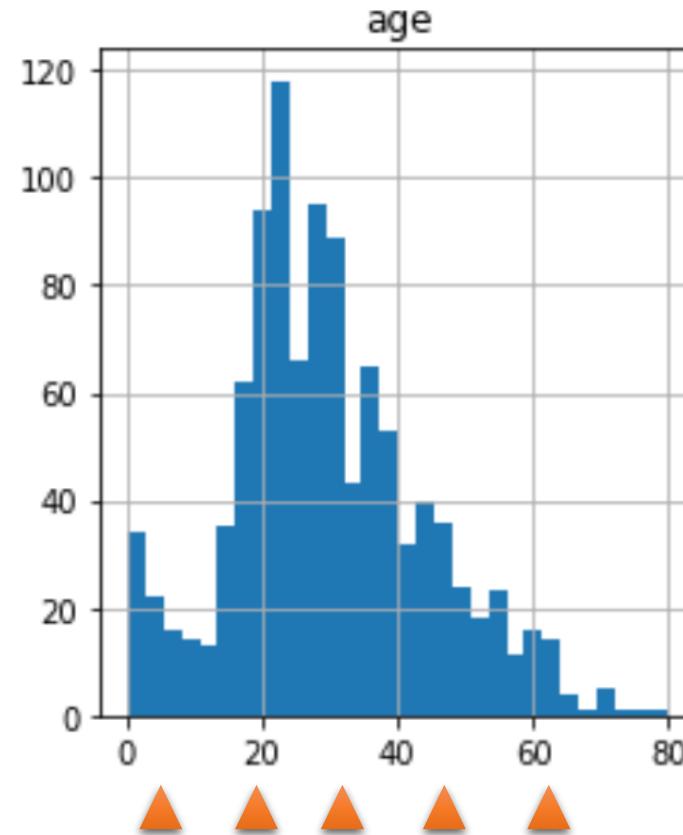
• K-means discretisation: definition

This discretisation method consists in applying k-means clustering to the continuous variable.

More details about k-means [here](#).



• K-means discretisation: calculation



Intervals = 5

Find 5 centroids

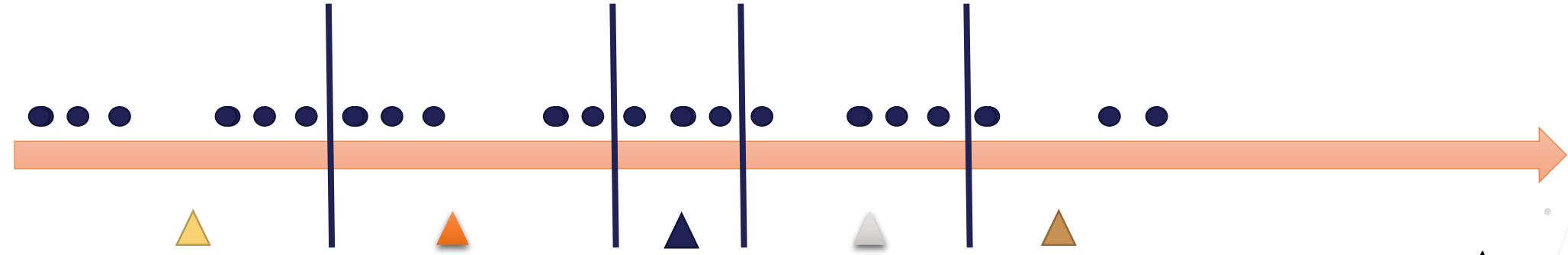
Divide data into clusters
according to distance to
centroids



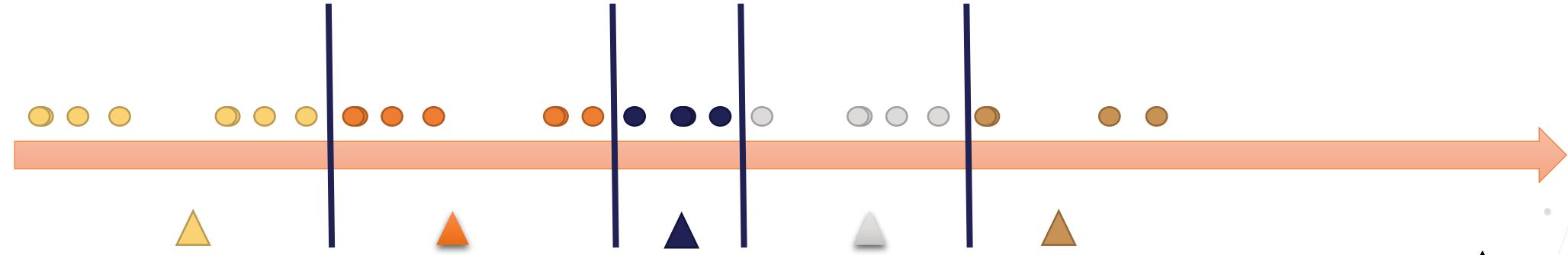
• K-means discretisation: calculation



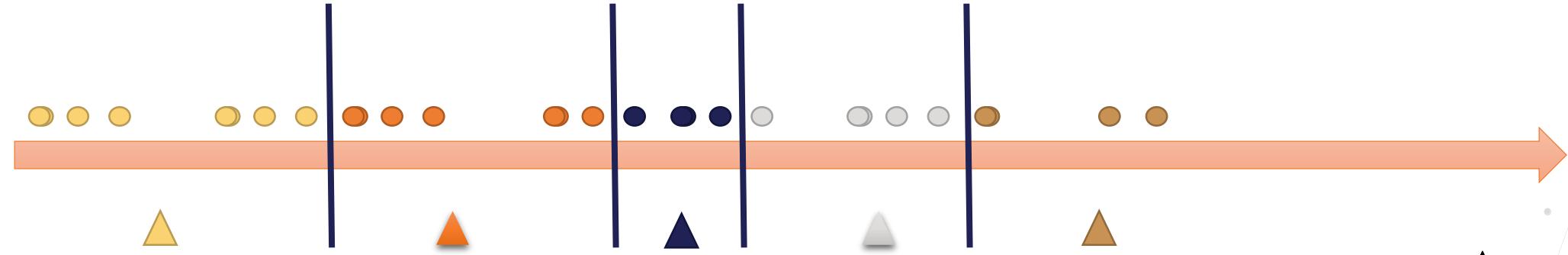
• K-means discretisation: calculation



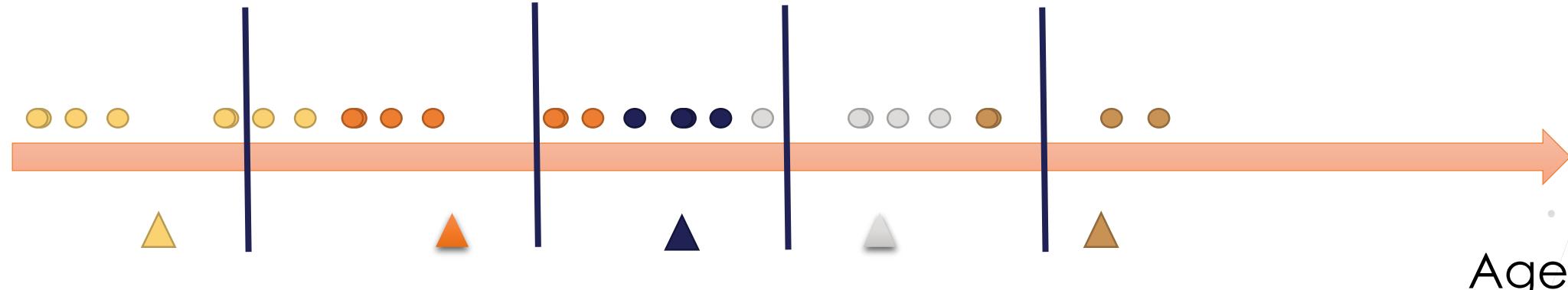
• K-means discretisation: calculation



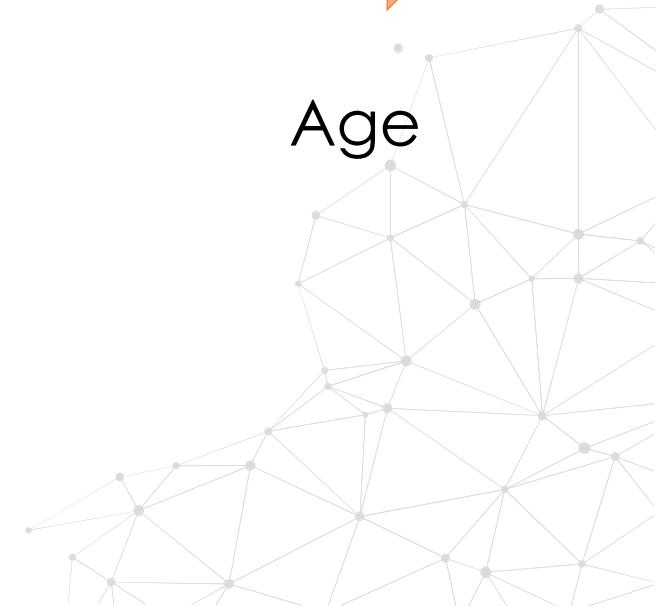
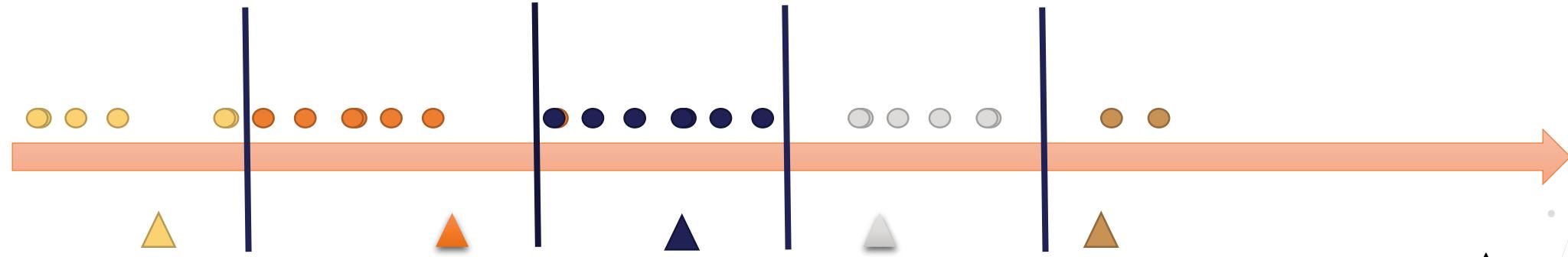
• K-means discretisation: calculation



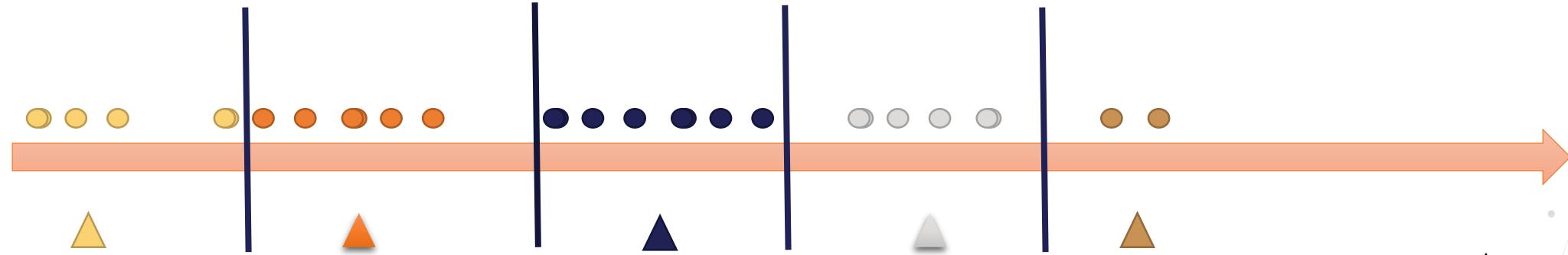
• K-means discretisation: calculation



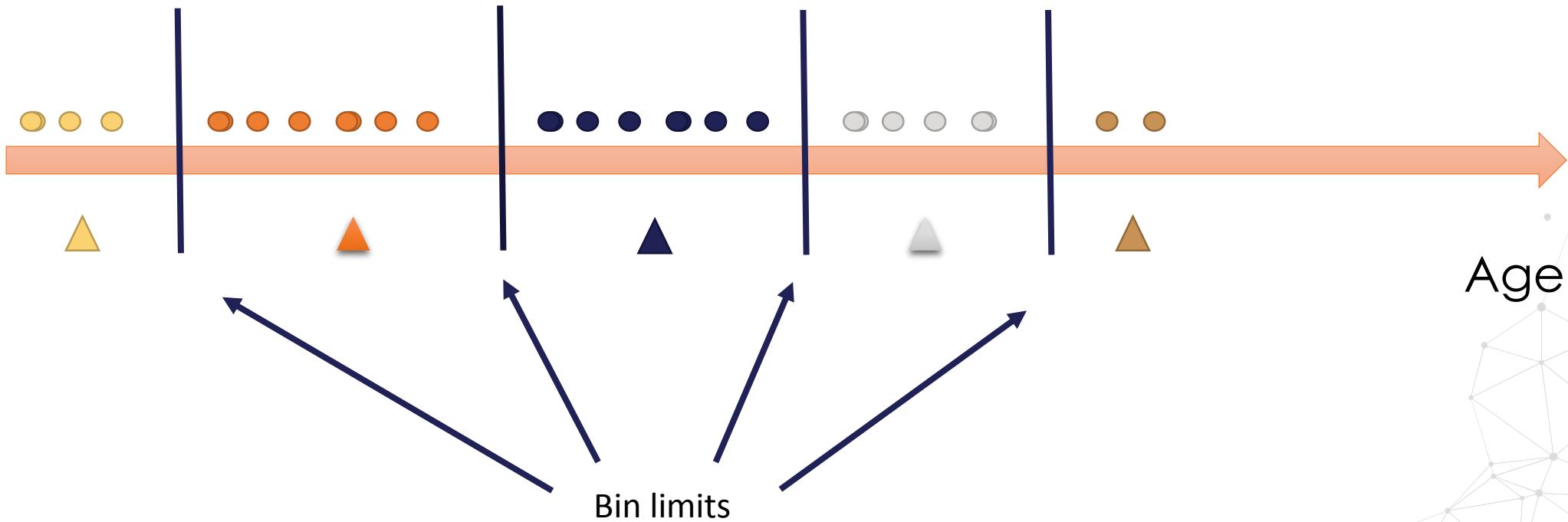
• K-means discretisation: calculation



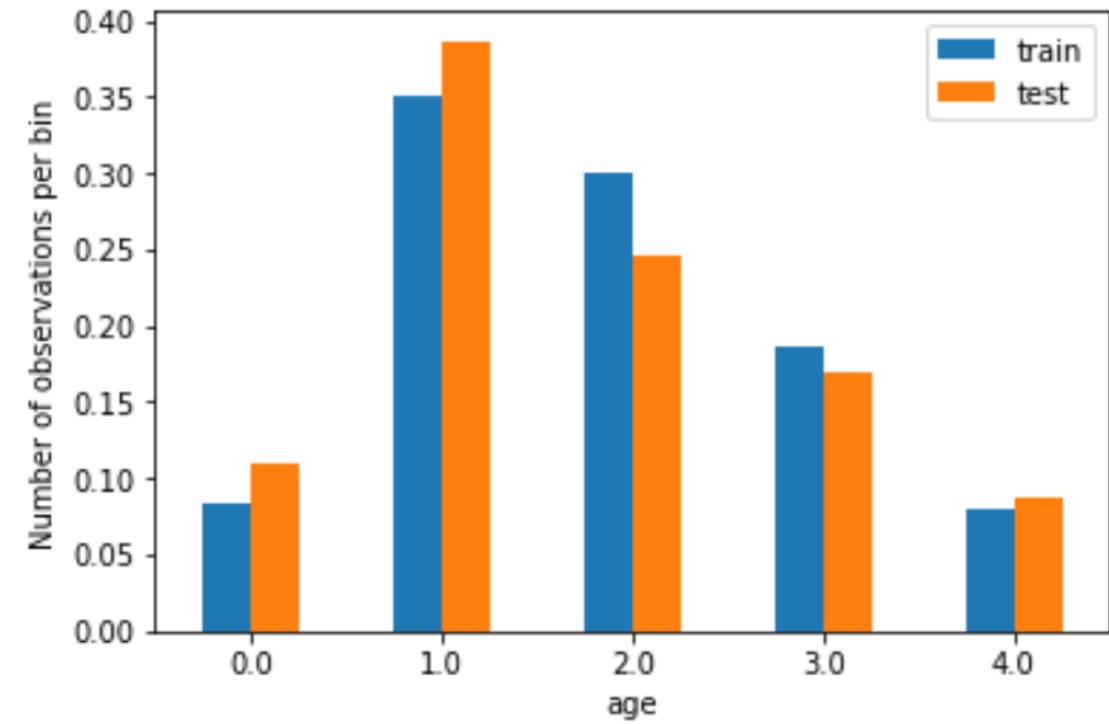
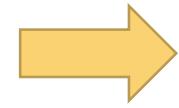
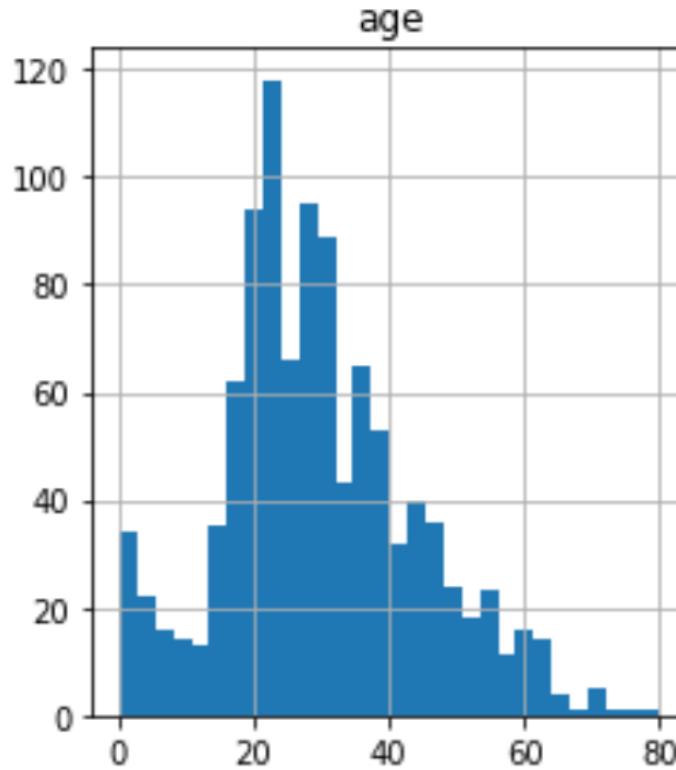
• K-means discretisation: calculation



• K-means discretisation: calculation



K-means discretisation: definition



• K-means discretisation: summary

- Does not improve value spread
- Handles outliers, although outliers may have an influence in the centroid
- Creates discrete variable
- Good to combine with categorical encodings





THANK YOU

www.trainindata.com

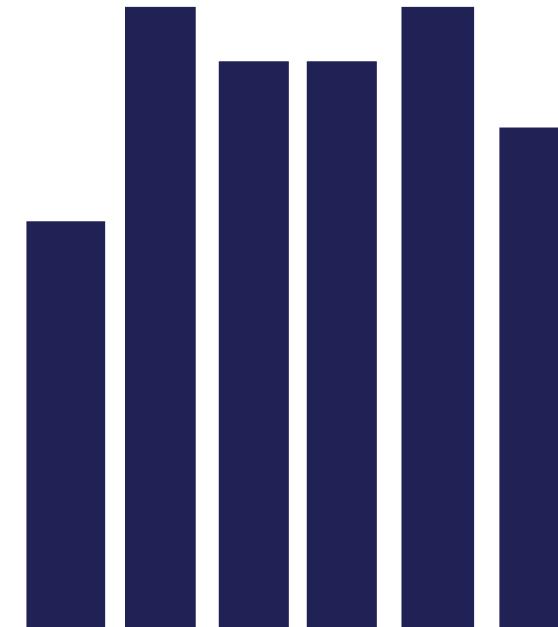
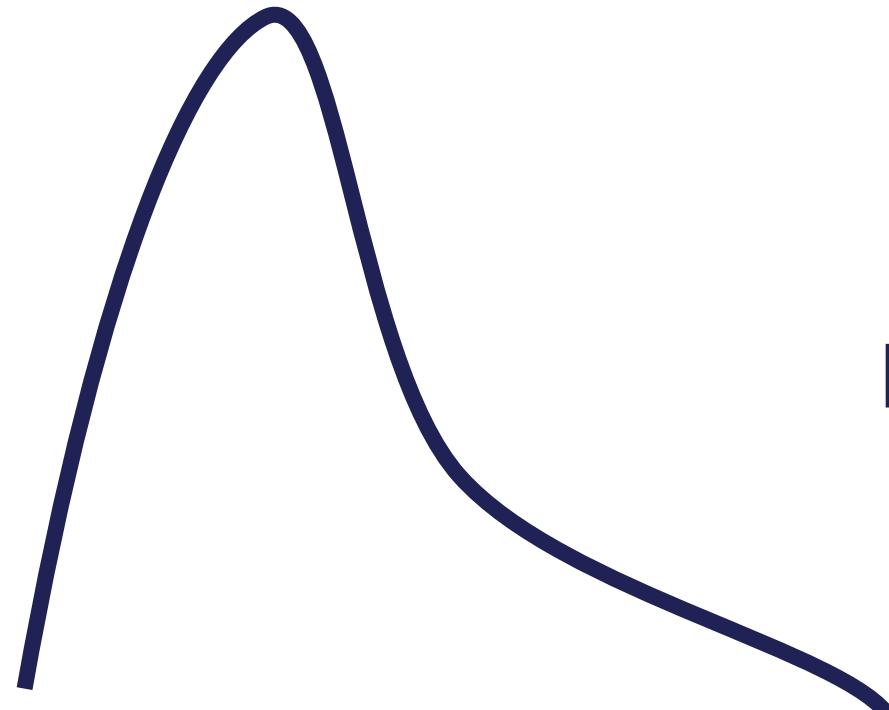




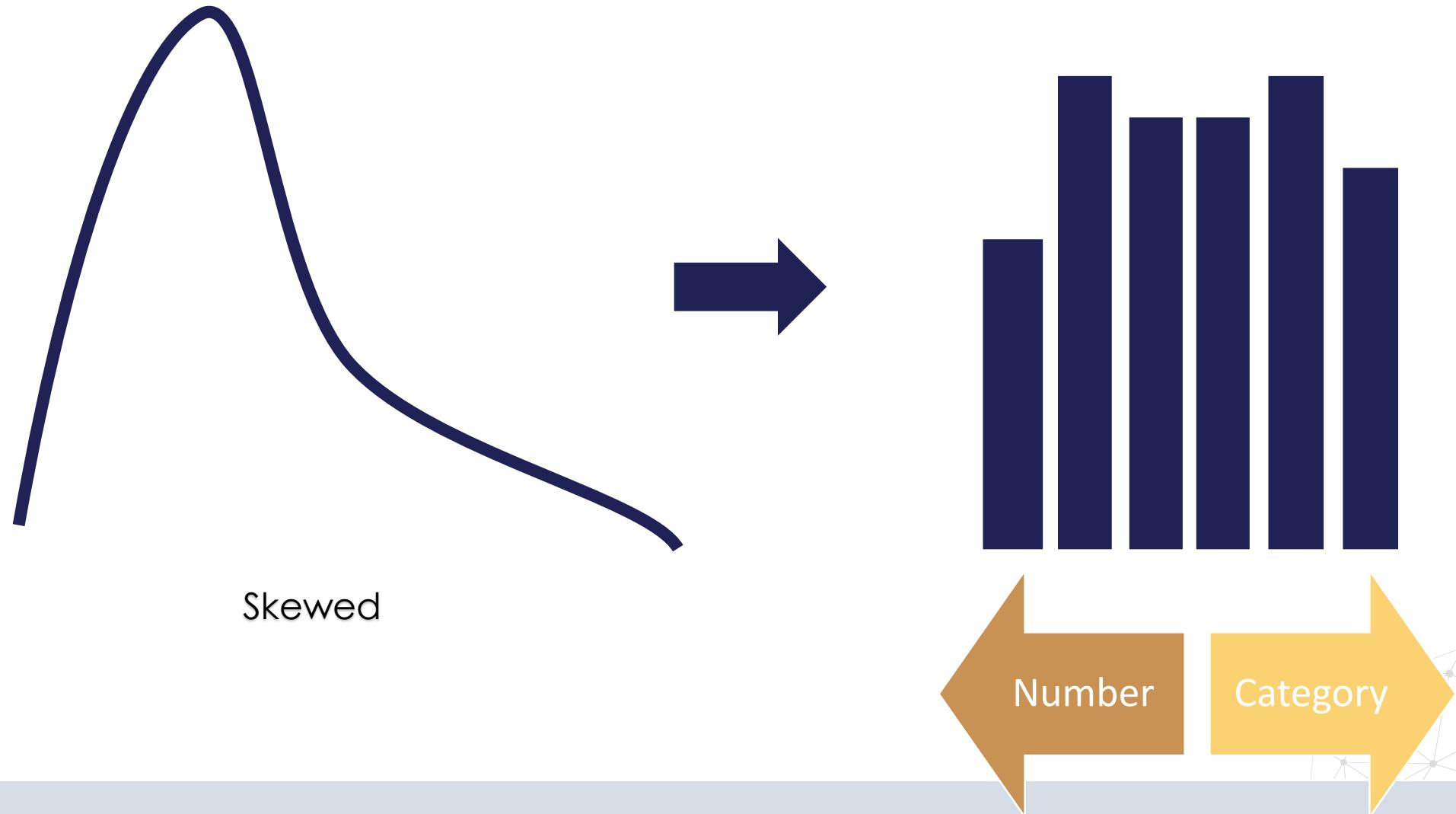
Train In Data

Discretisation plus encoding

Discretisation

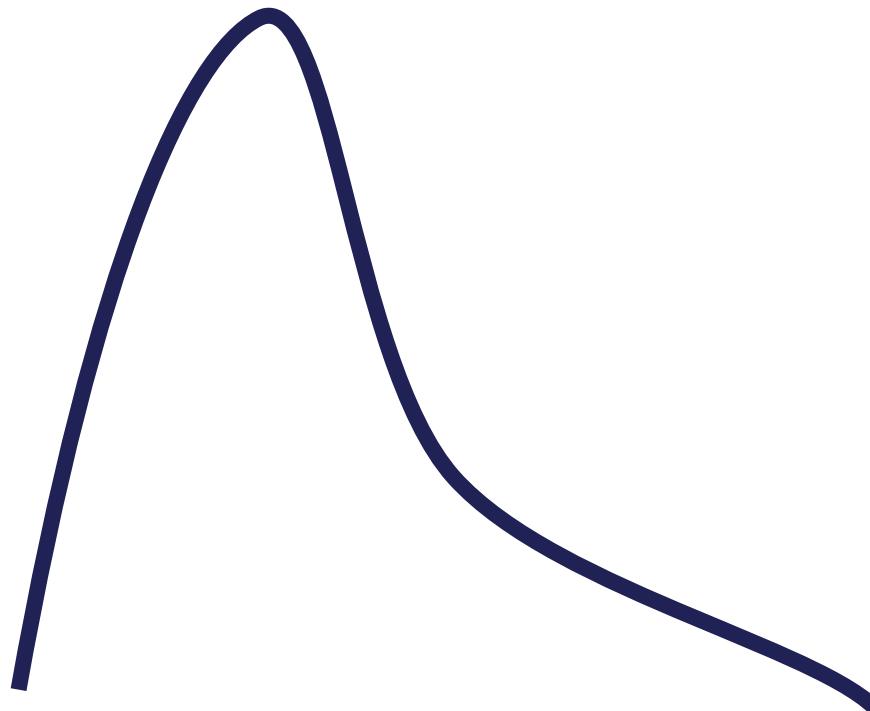


Discretisation: What next?

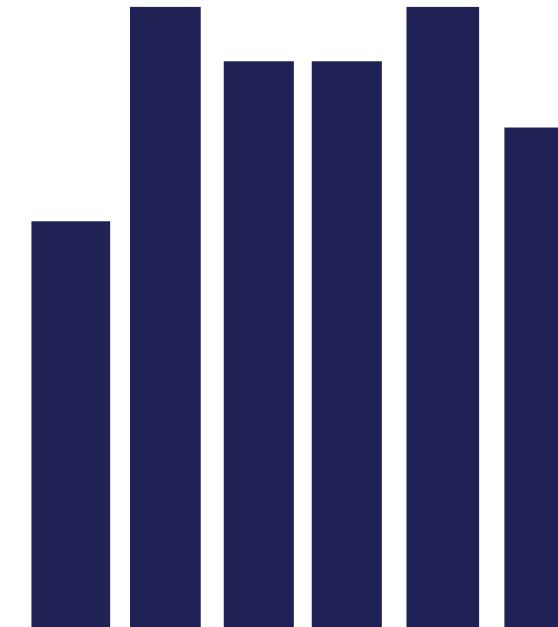


Discretisation plus encoding

bin = category



Skewed

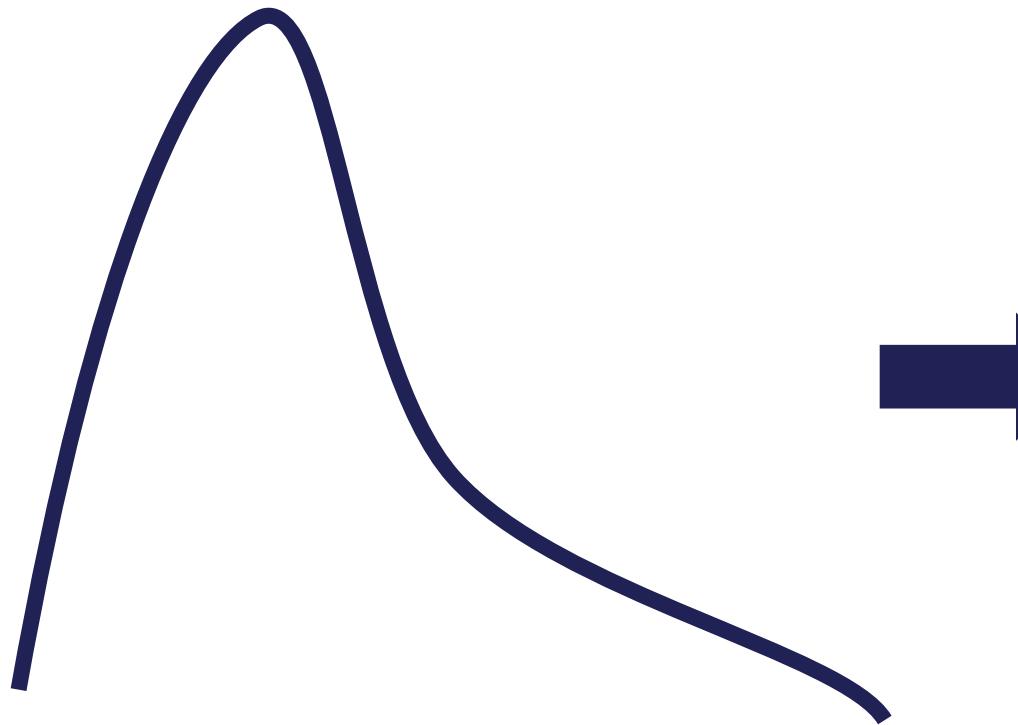


Discrete

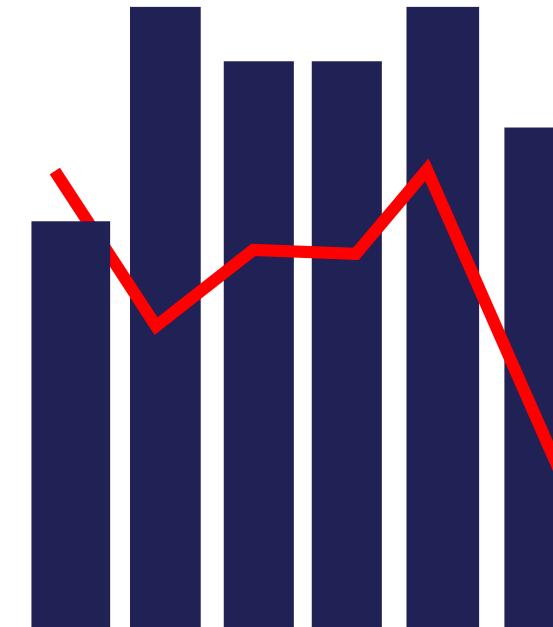


Discretisation plus encoding

bin = category



Skewed

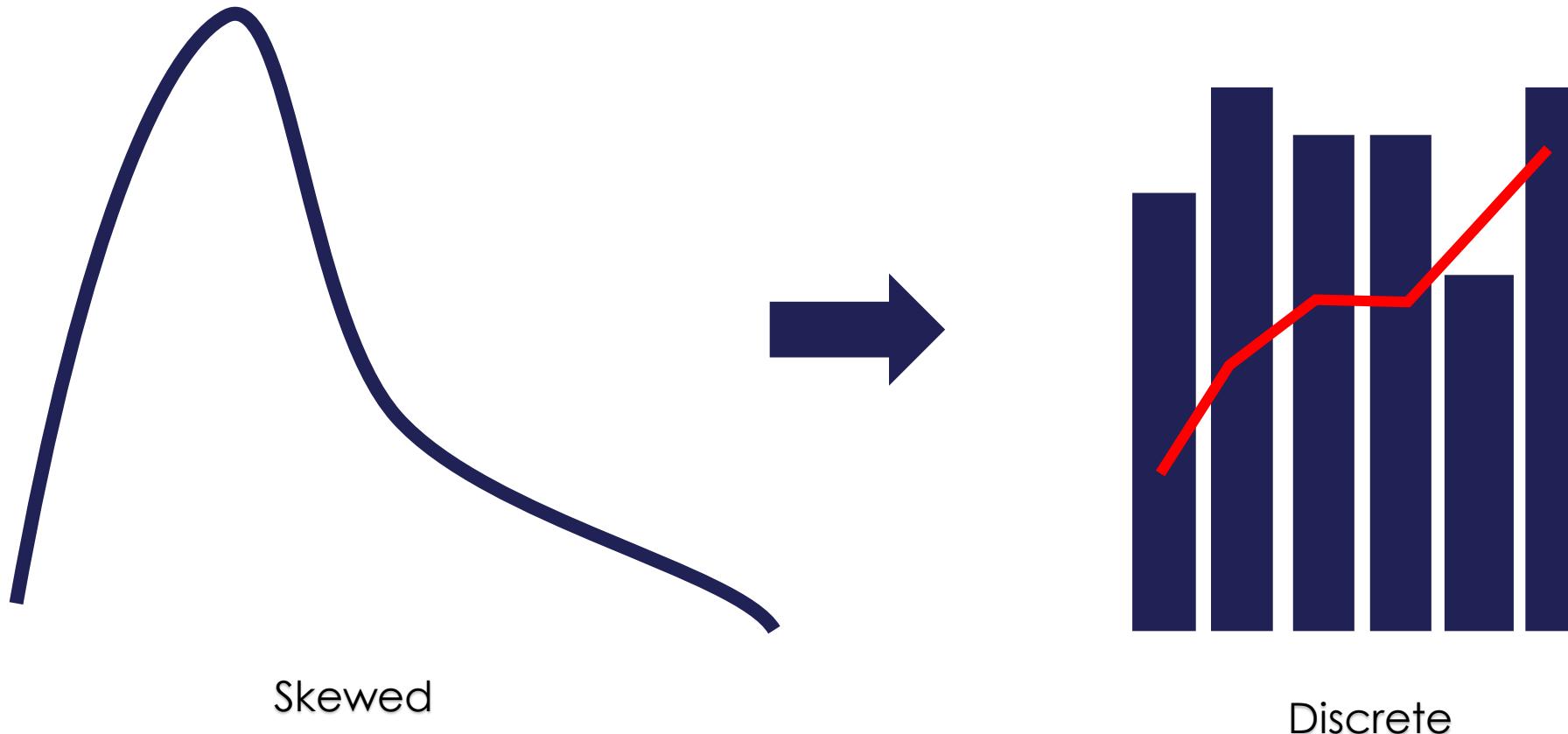


Discrete

target



Discretisation plus encoding



Target
**Monotonic
encoding**



THANK YOU

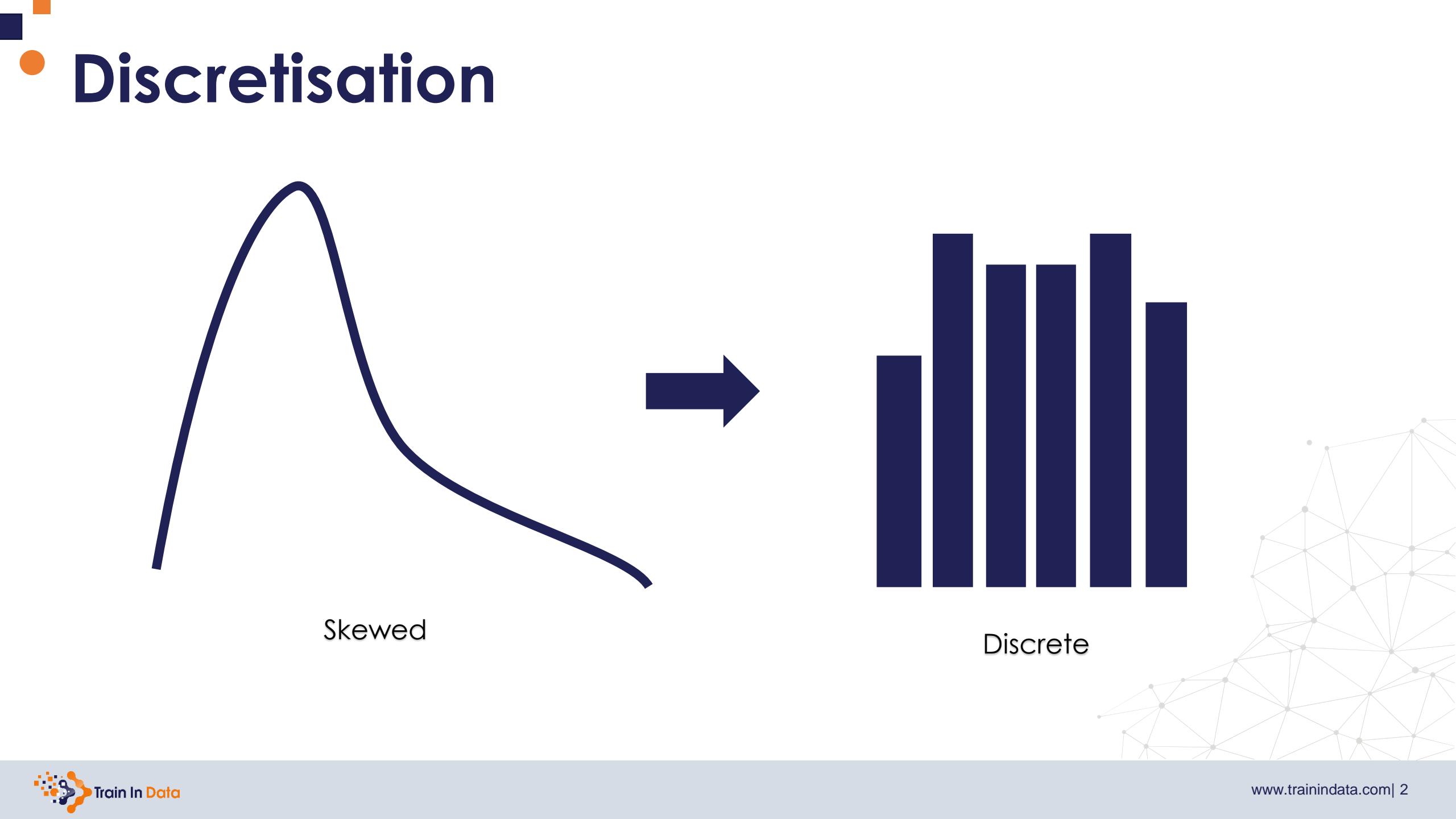
www.trainindata.com



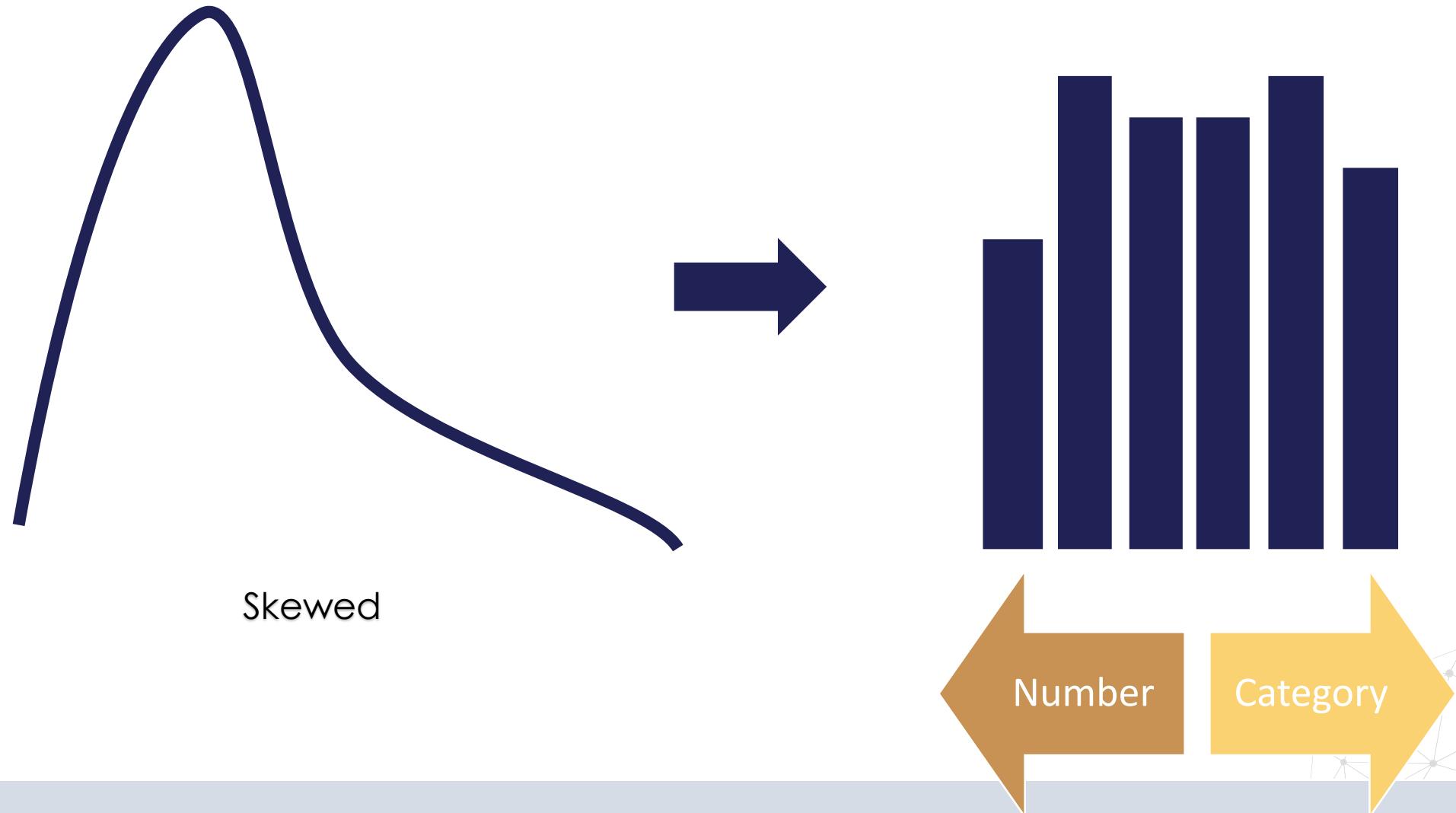


Train In Data

Discretisation plus encoding

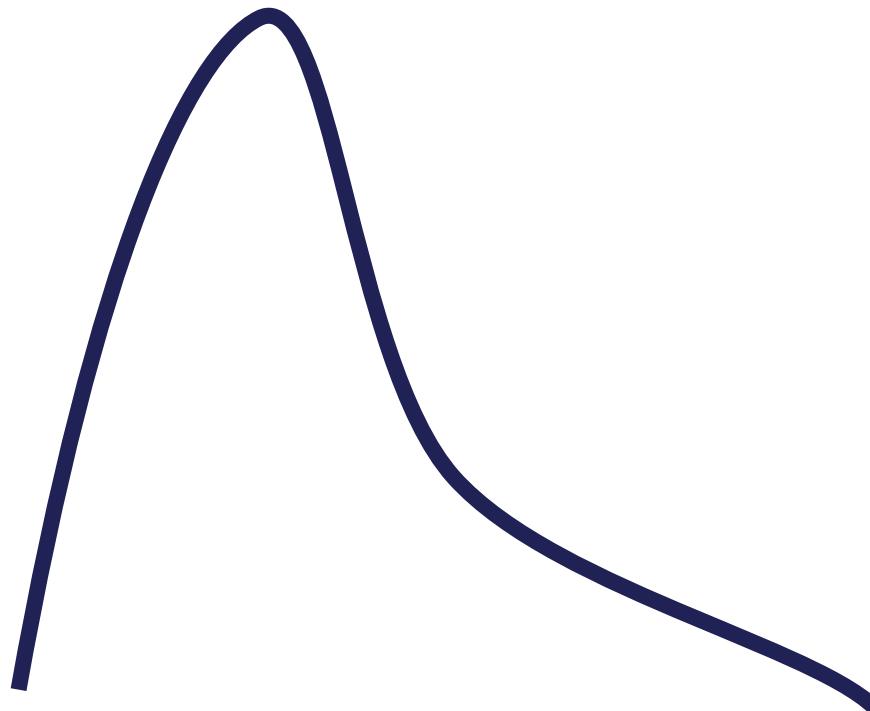


Discretisation: What next?

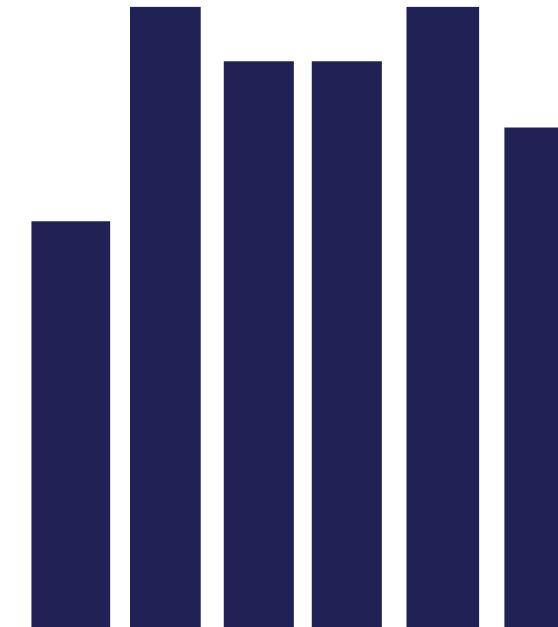


Discretisation plus encoding

bin = category



Skewed

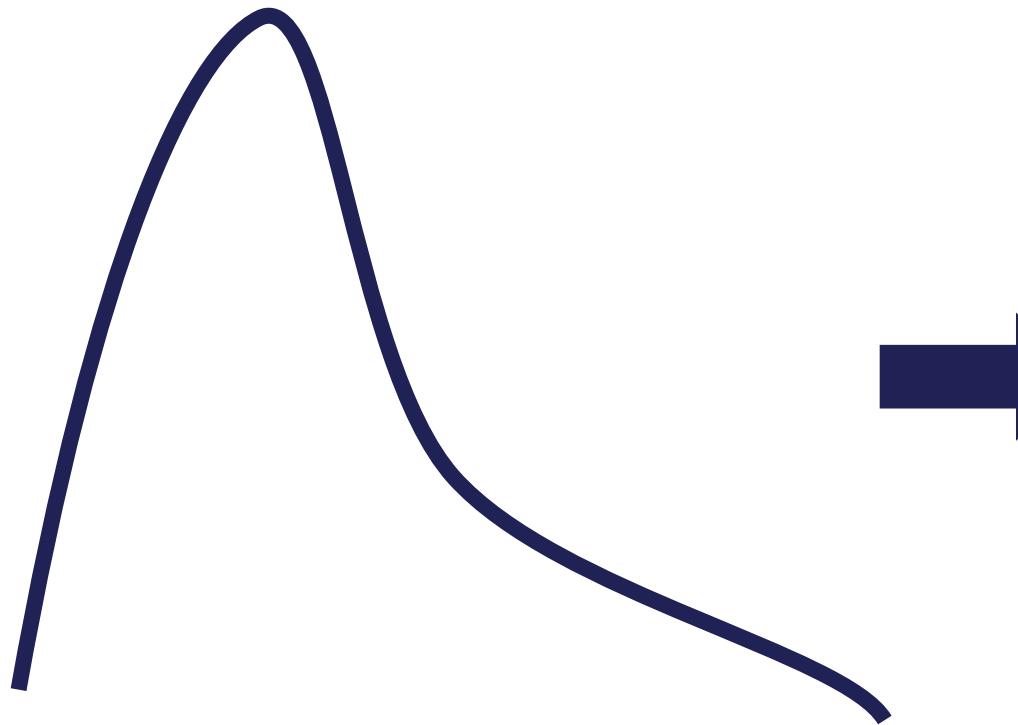


Discrete

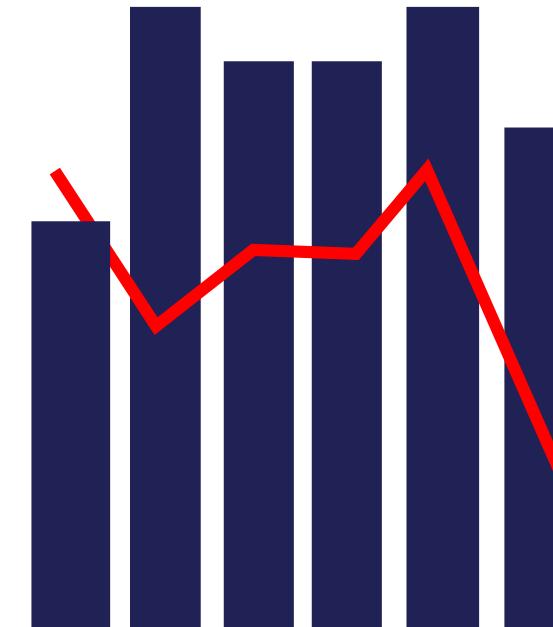


Discretisation plus encoding

bin = category



Skewed

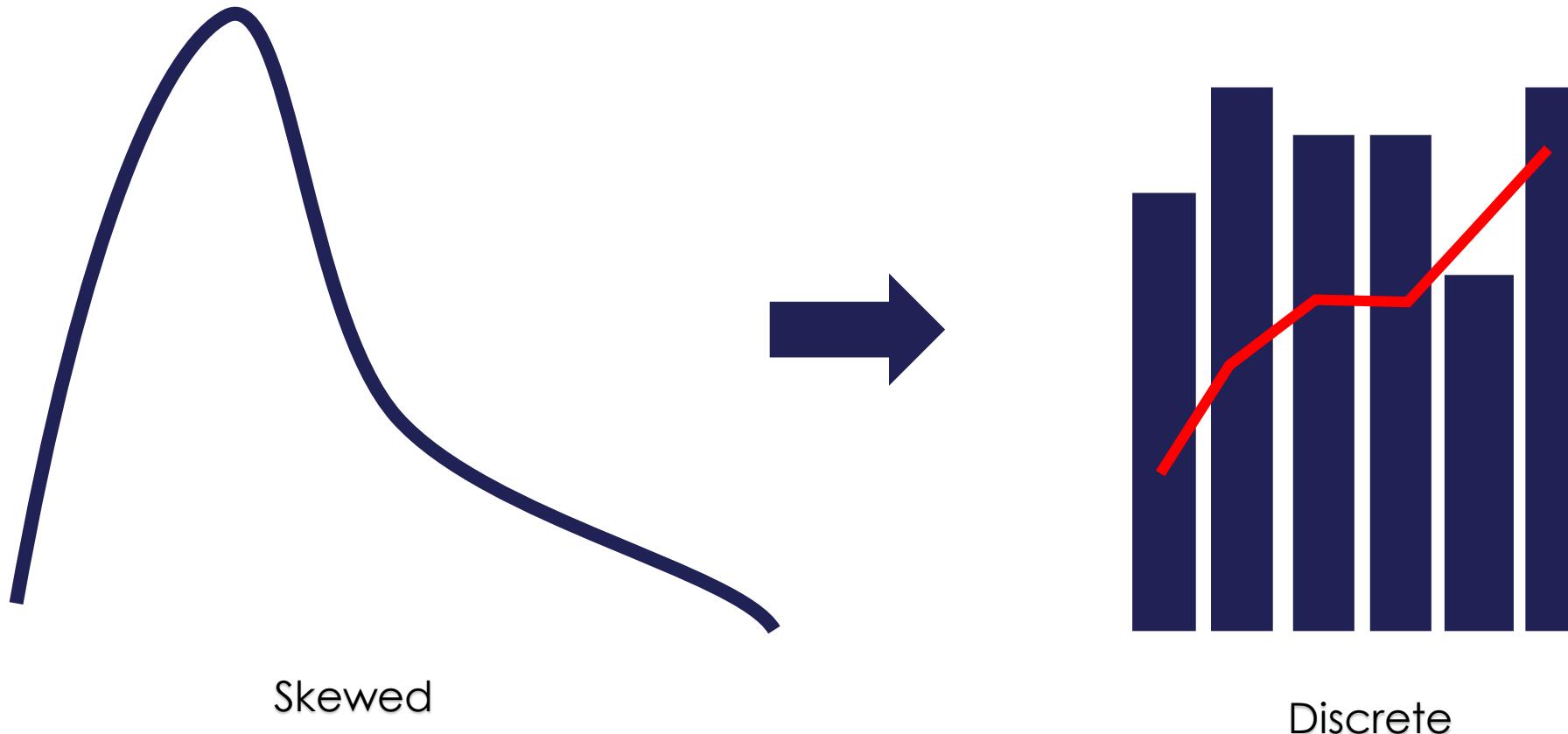


Discrete

target



Discretisation plus encoding



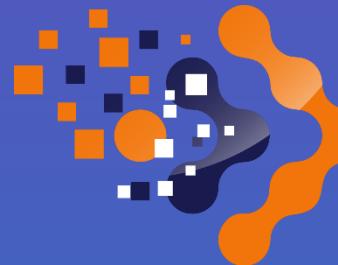
Target
**Monotonic
encoding**



THANK YOU

www.trainindata.com





Train In Data

Discretisation with decision trees

• Discretisation with decision trees

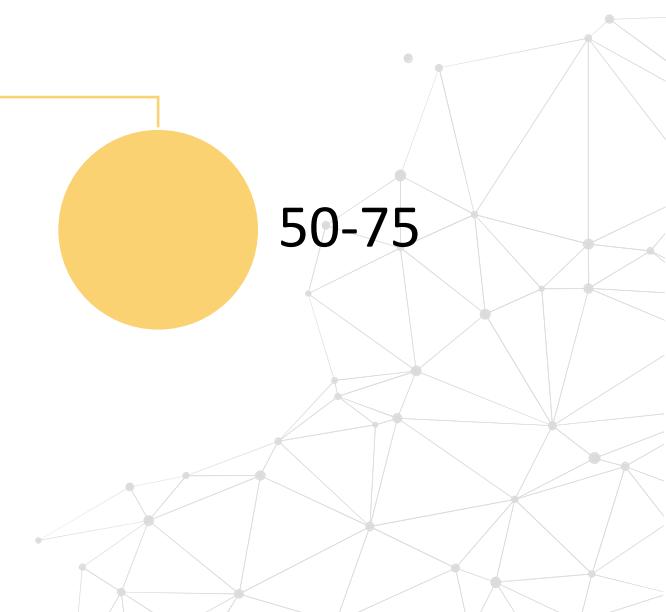
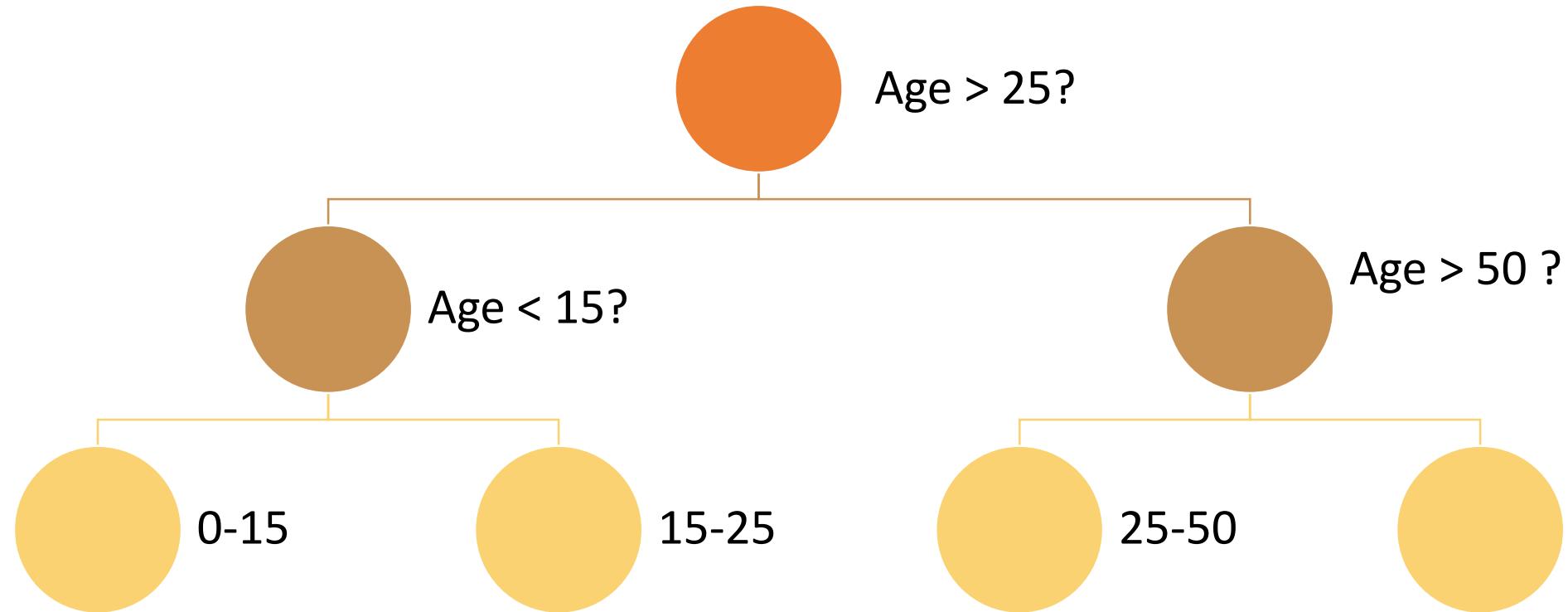
Discretisation with Decision Trees consists in using a decision tree to identify the optimal bins.

When a decision tree makes a decision, it assigns an observation to one of n end leaves.

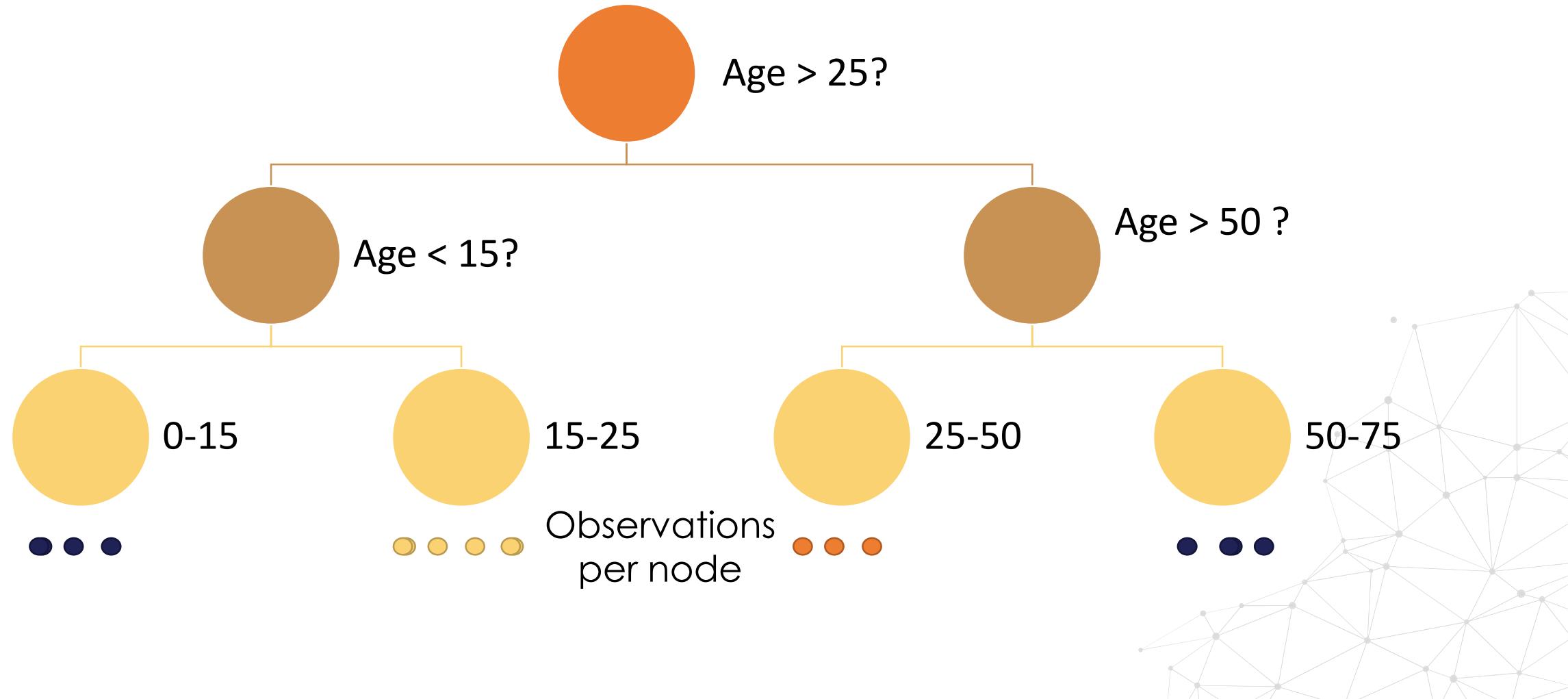
Decision tree creates a discrete output, which values are the predictions at each of its n leaves.



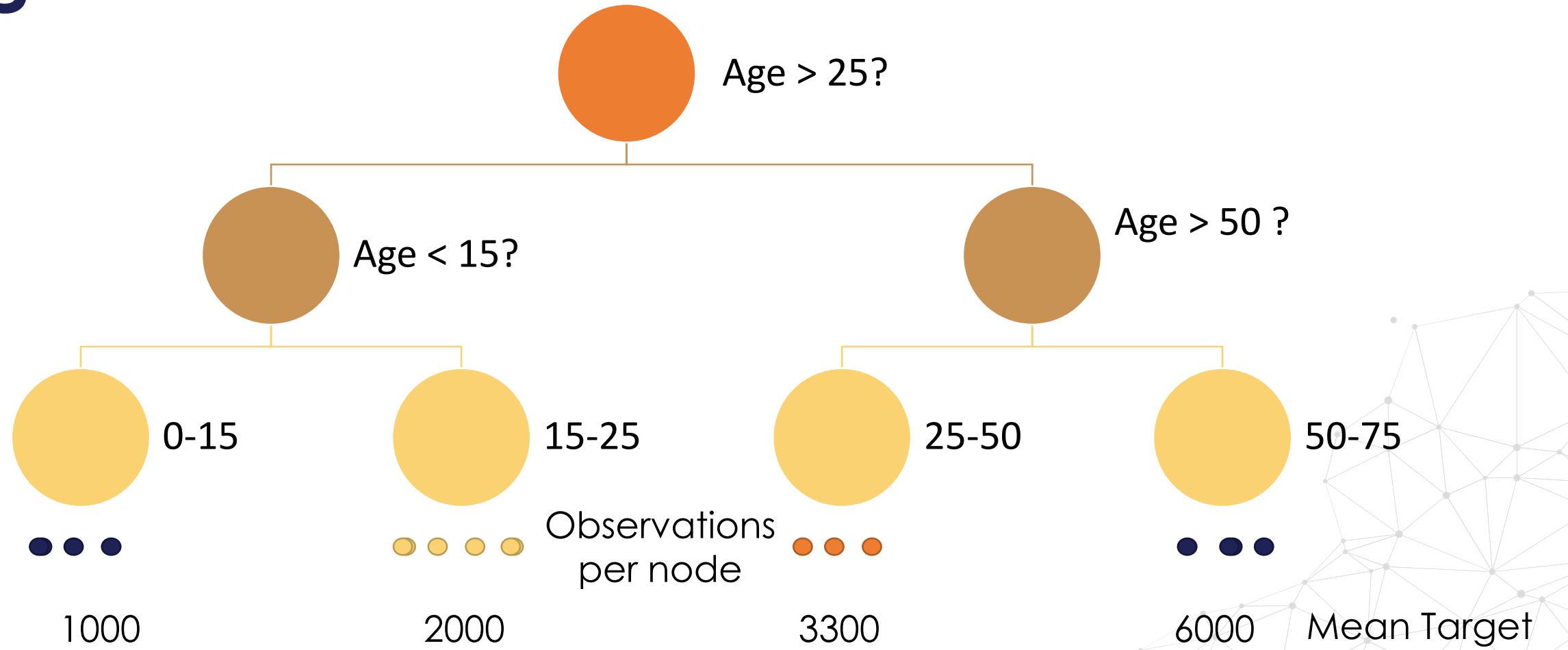
Discretisation with decision trees



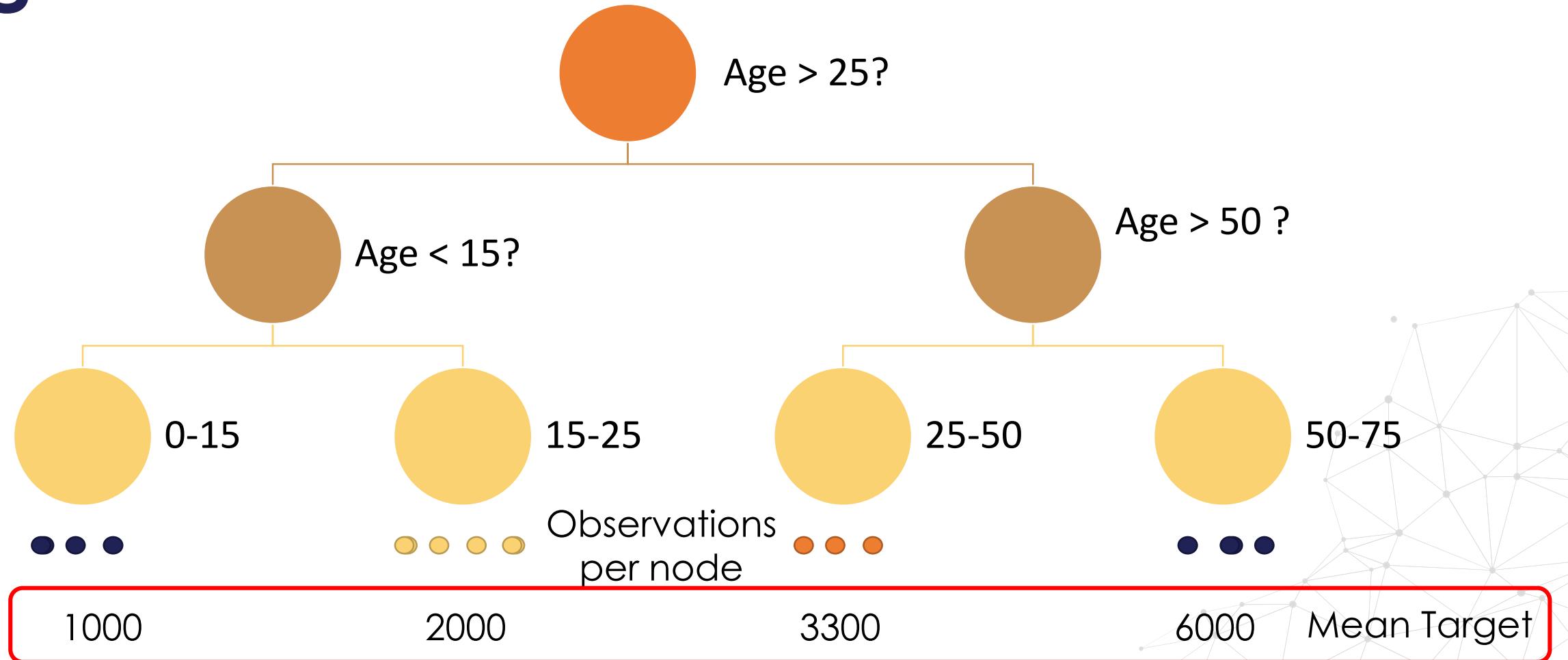
Discretisation with decision trees



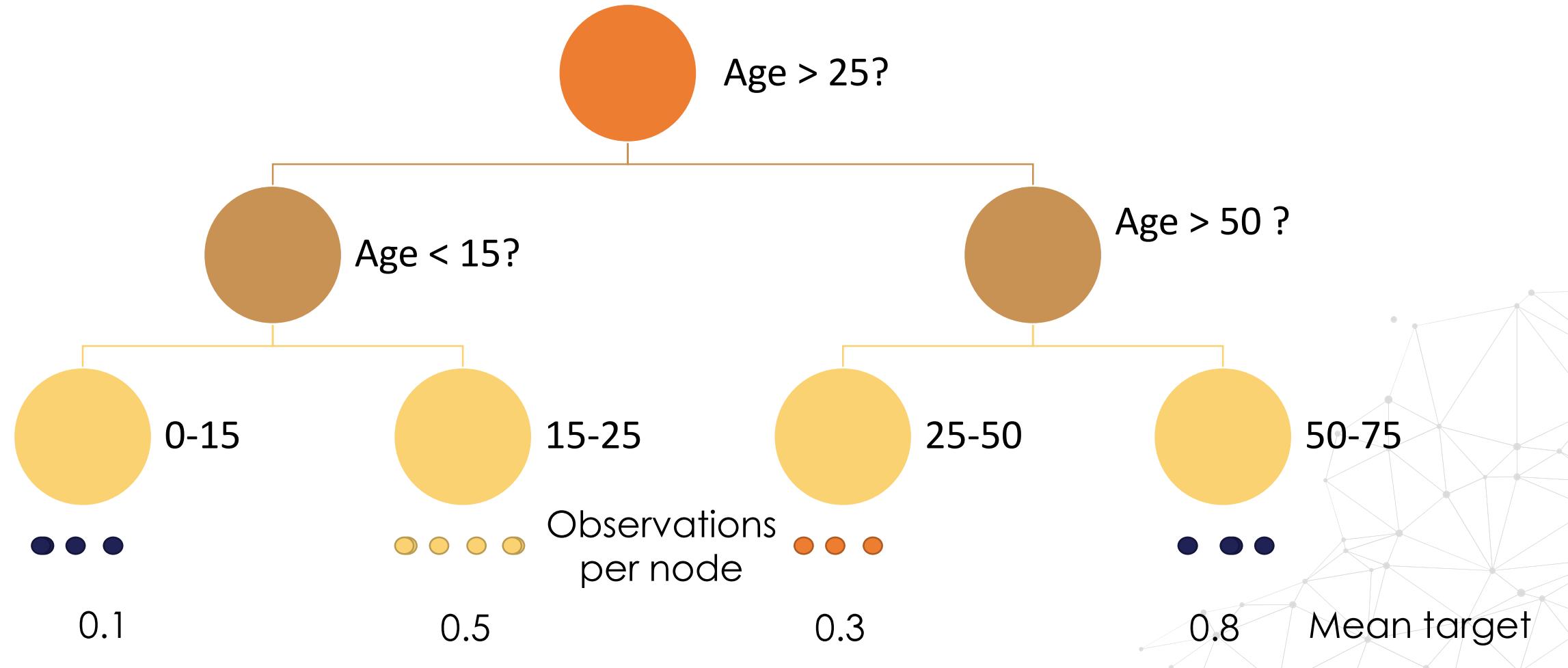
Discretisation with decision trees: regression



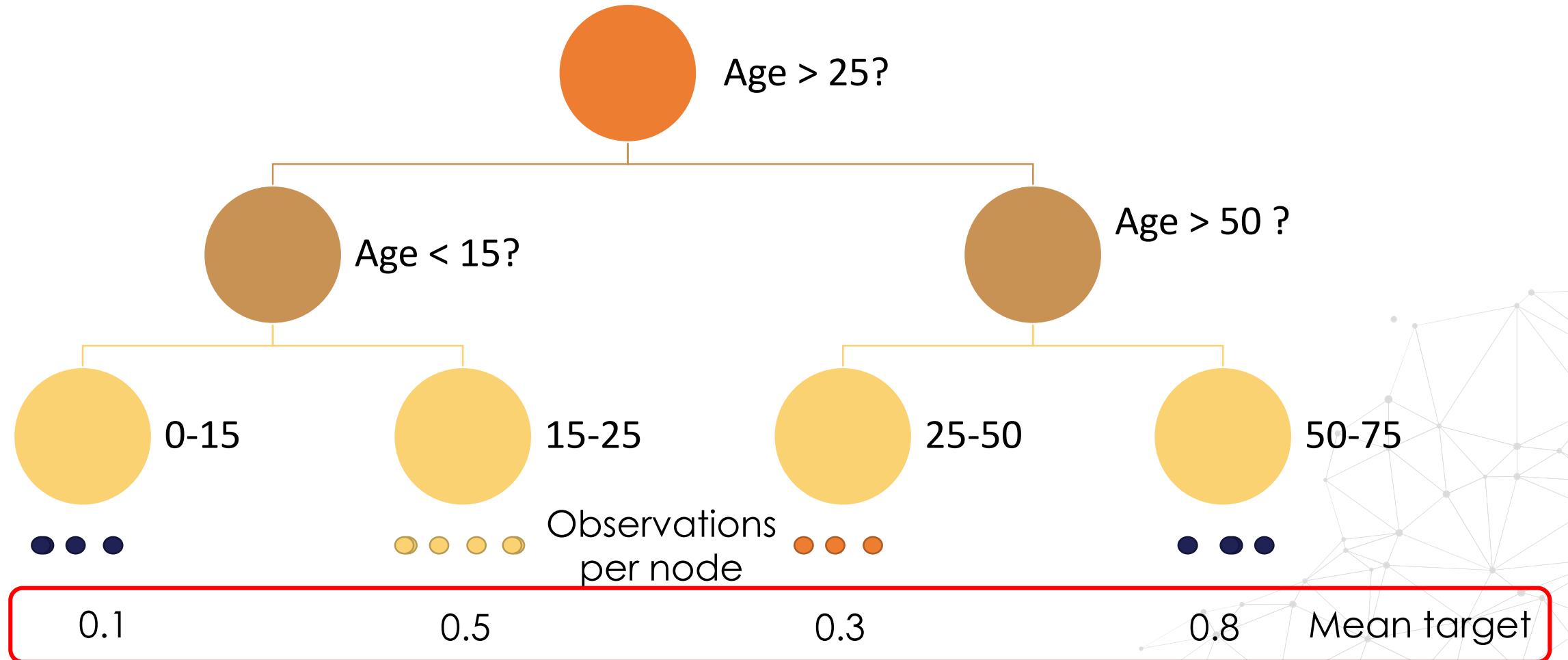
Discretisation with decision trees: regression



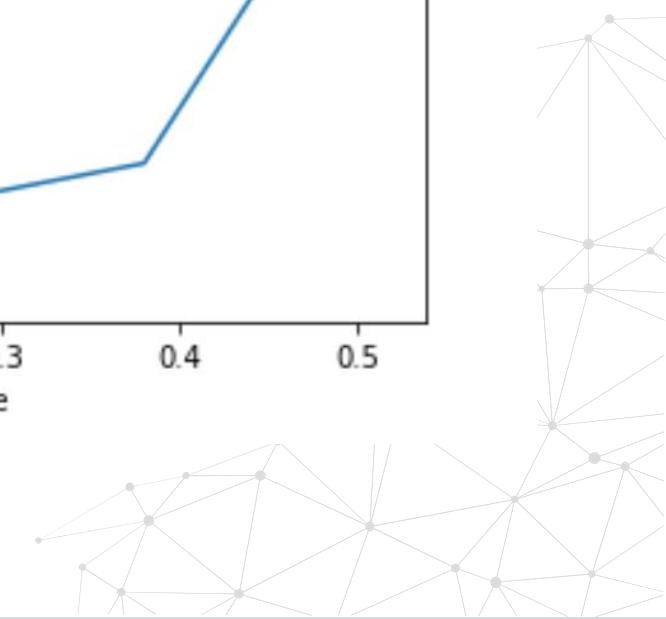
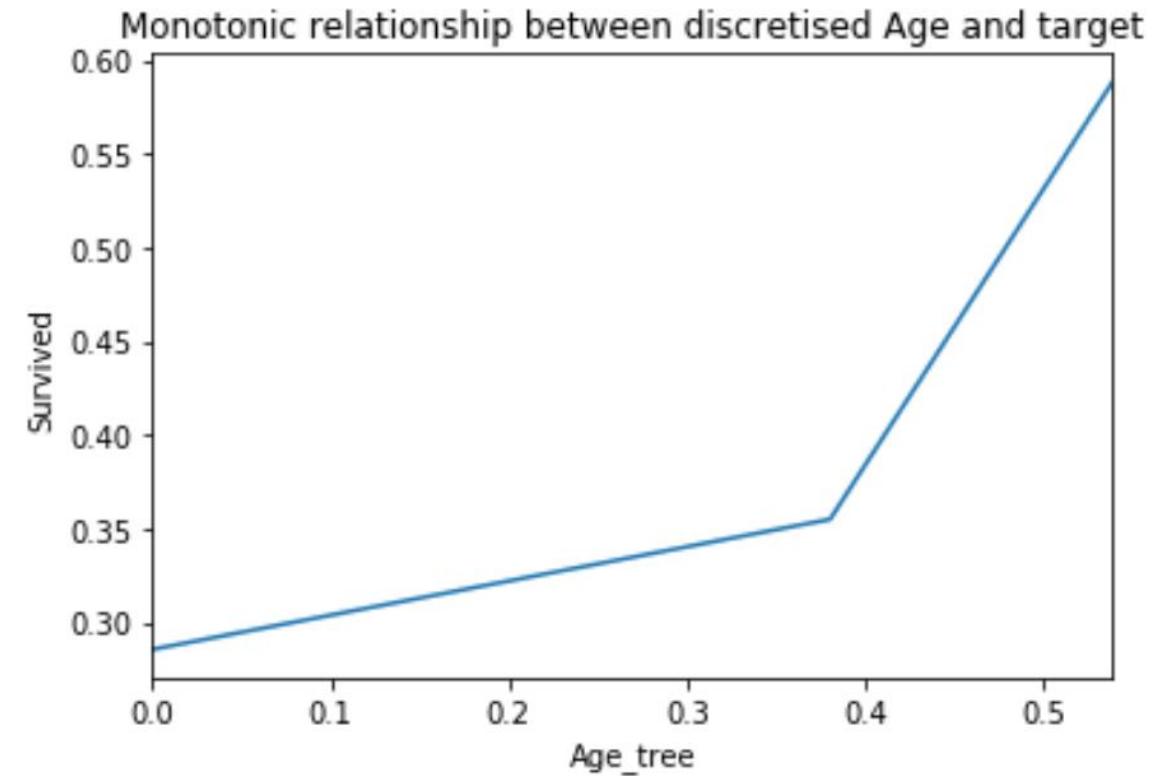
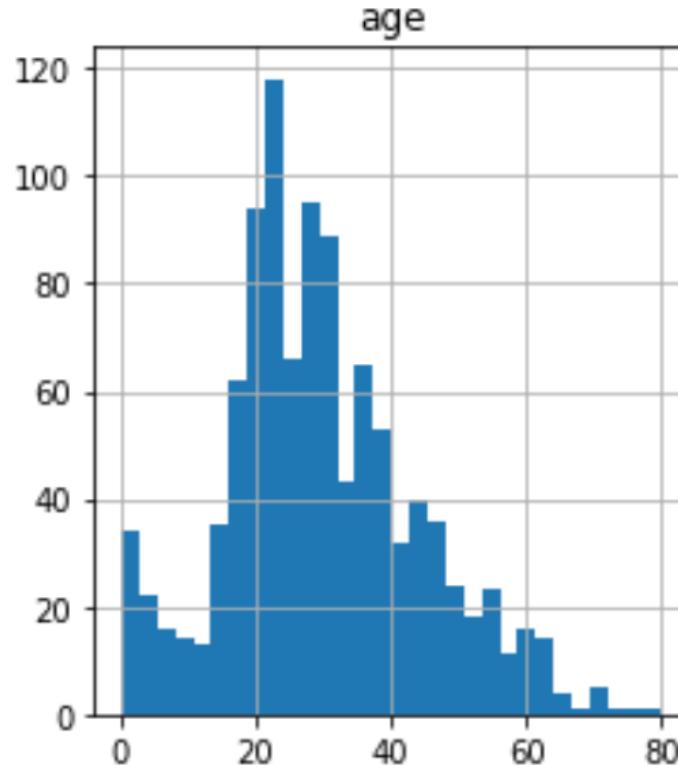
Discretisation with decision trees: classification



Discretisation with decision trees: classification



Discretisation with decision trees



• Discretisation with decision trees: summary

- Does not improve value spread
- Handles outliers, trees are robust to outliers
- Creates discrete variable
- Creates monotonic relationship





THANK YOU

www.trainindata.com





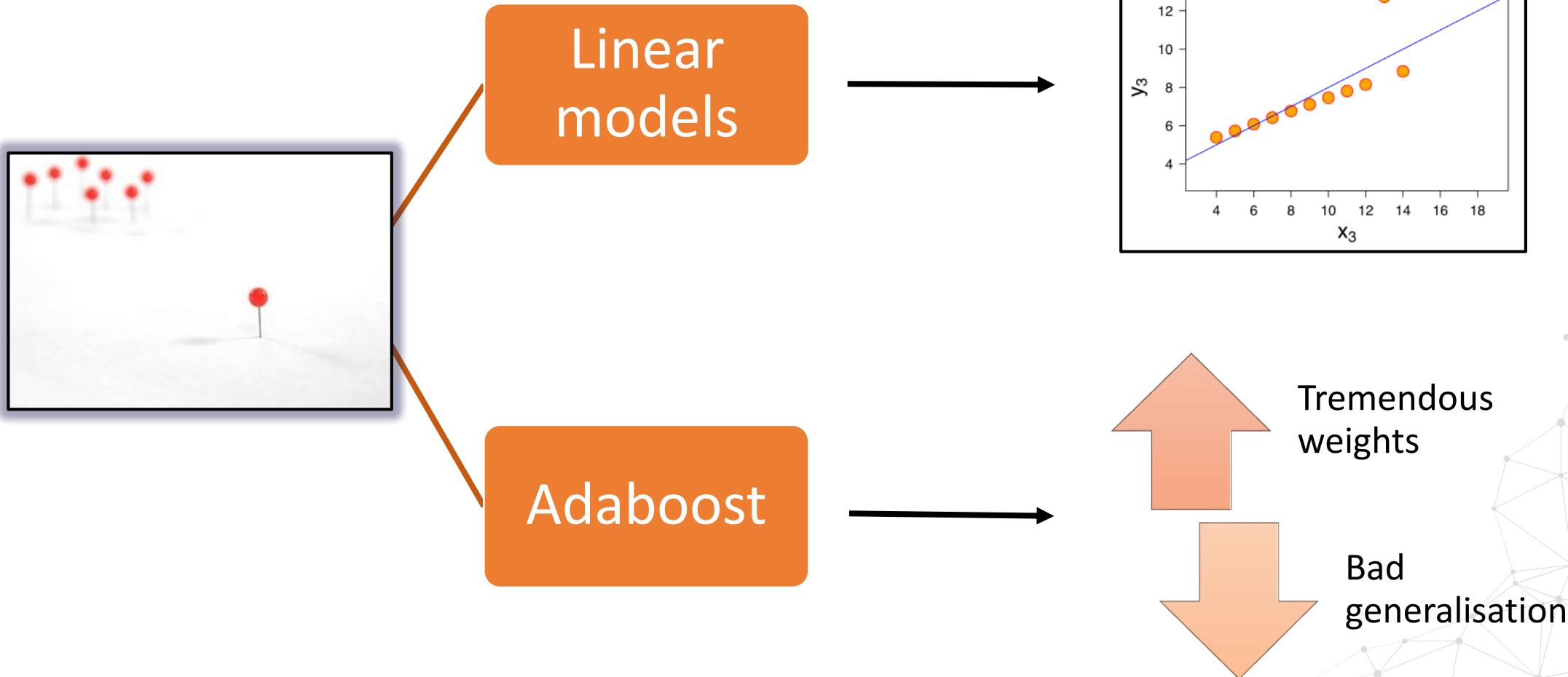
Outlier Engineering

Outliers

- An outlier is a data point which is significantly different from the remaining data.
- “An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.” [D. Hawkins. Identification of Outliers, Chapman and Hall , 1980.]



Algorithms susceptible to outliers





Handling Outliers

What can we do if we find outliers in our variables?



• Ways to engineer outliers

Trimming

- Removing outliers from the data set

Missing data

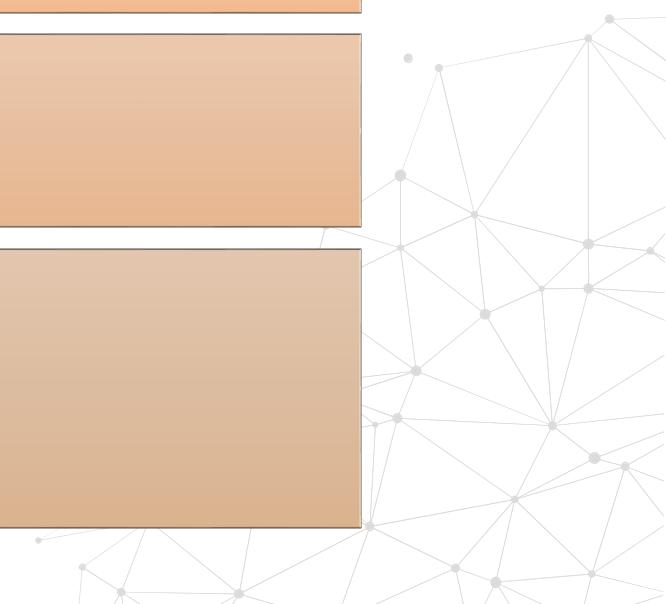
- Treat outliers as missing data and perform missing data imputation
- Any technique in section 4

Discretisation

- Put outliers into lower / upper bins
- Any technique in section 8

Censoring

- Capping
- Top / Bottom coding
- Winsorization



• Ways to engineer outliers

Trimming

- Removing outliers from the data set

Missing data

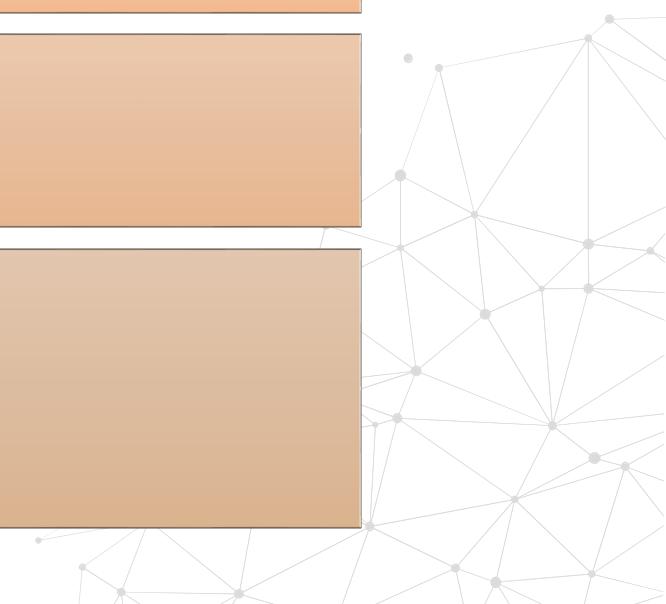
- Treat outliers as missing data and perform missing data imputation
- **Any technique in section 4**

Discretisation

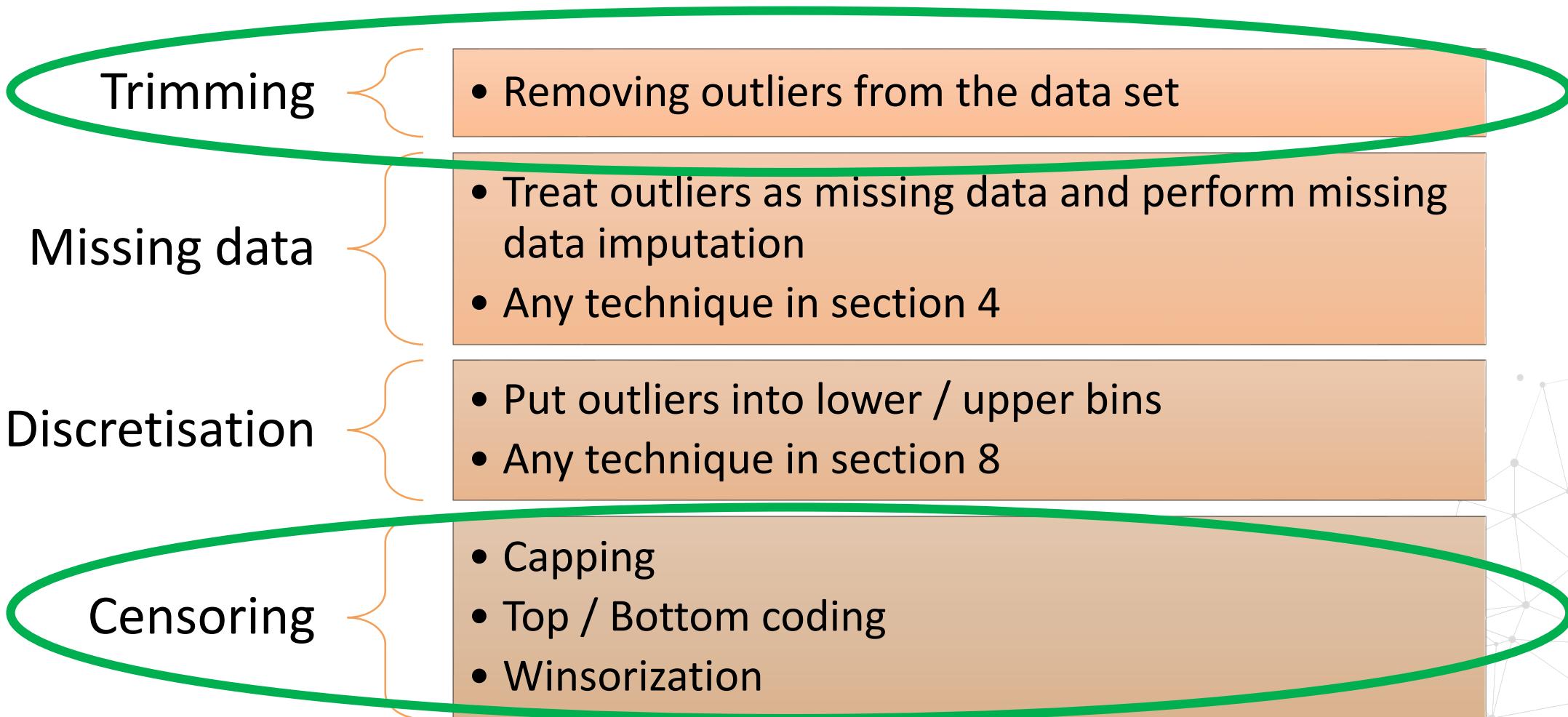
- Put outliers into lower / upper bins
- **Any technique in section 8**

Censoring

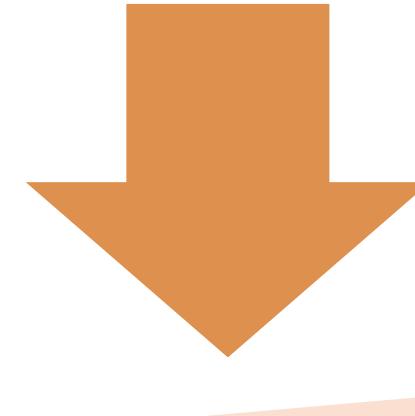
- Capping
- Top / Bottom coding
- Winsorization



Ways to engineer outliers

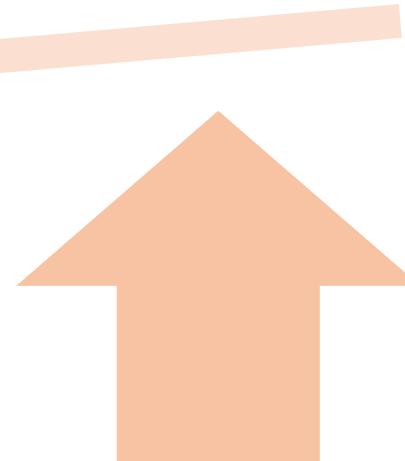


Trimming: pros and cons

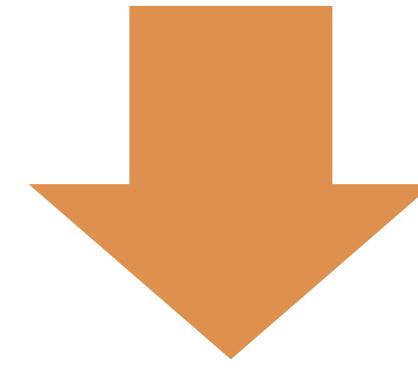


Can remove
big chunk of
data

Fast

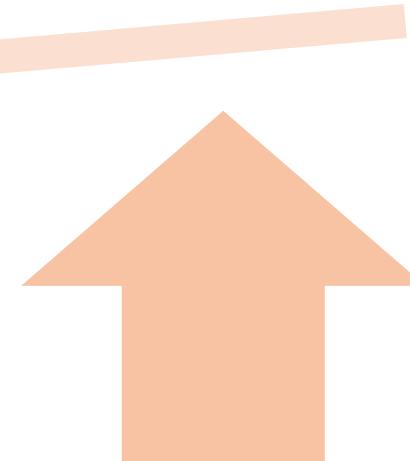


Capping: pros and cons



No data is removed

Distorts variable distribution





Detecting Outliers

Extreme Value Analysis

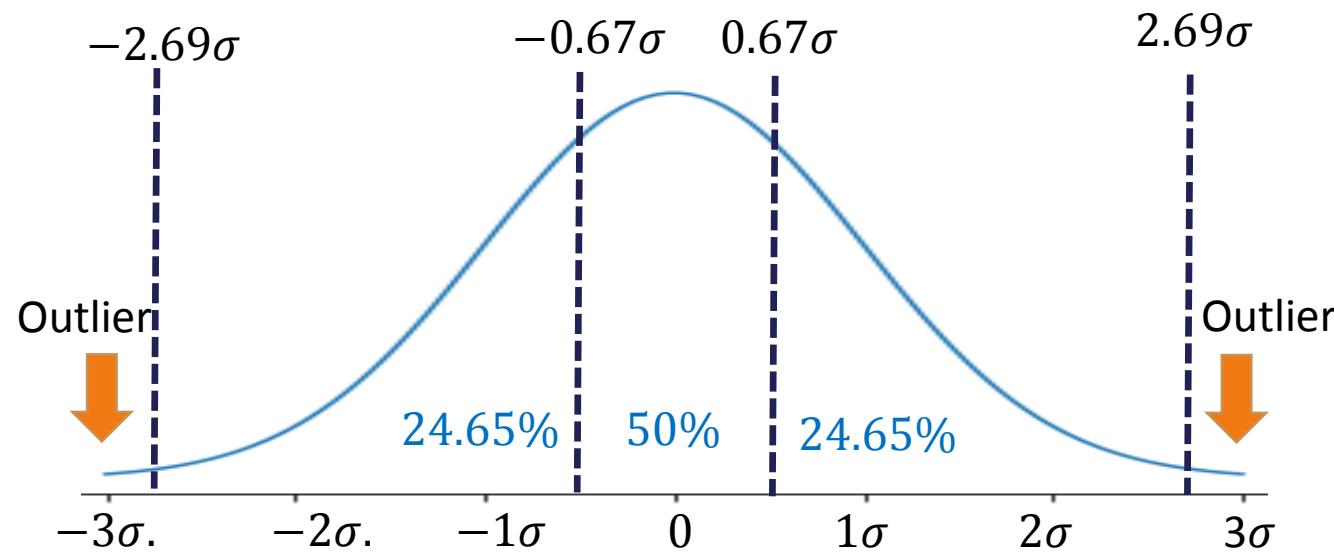


• Detecting outliers

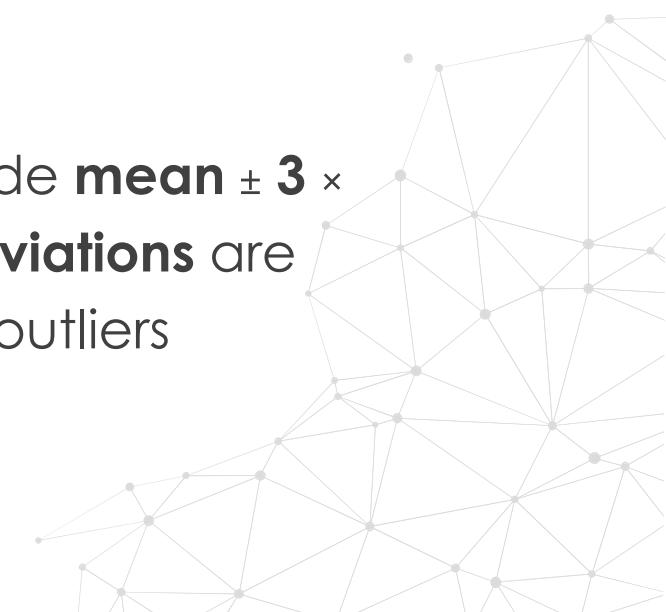
- Gaussian distribution (mean and std)
- Inter-quantal range proximity rule
- Quantiles



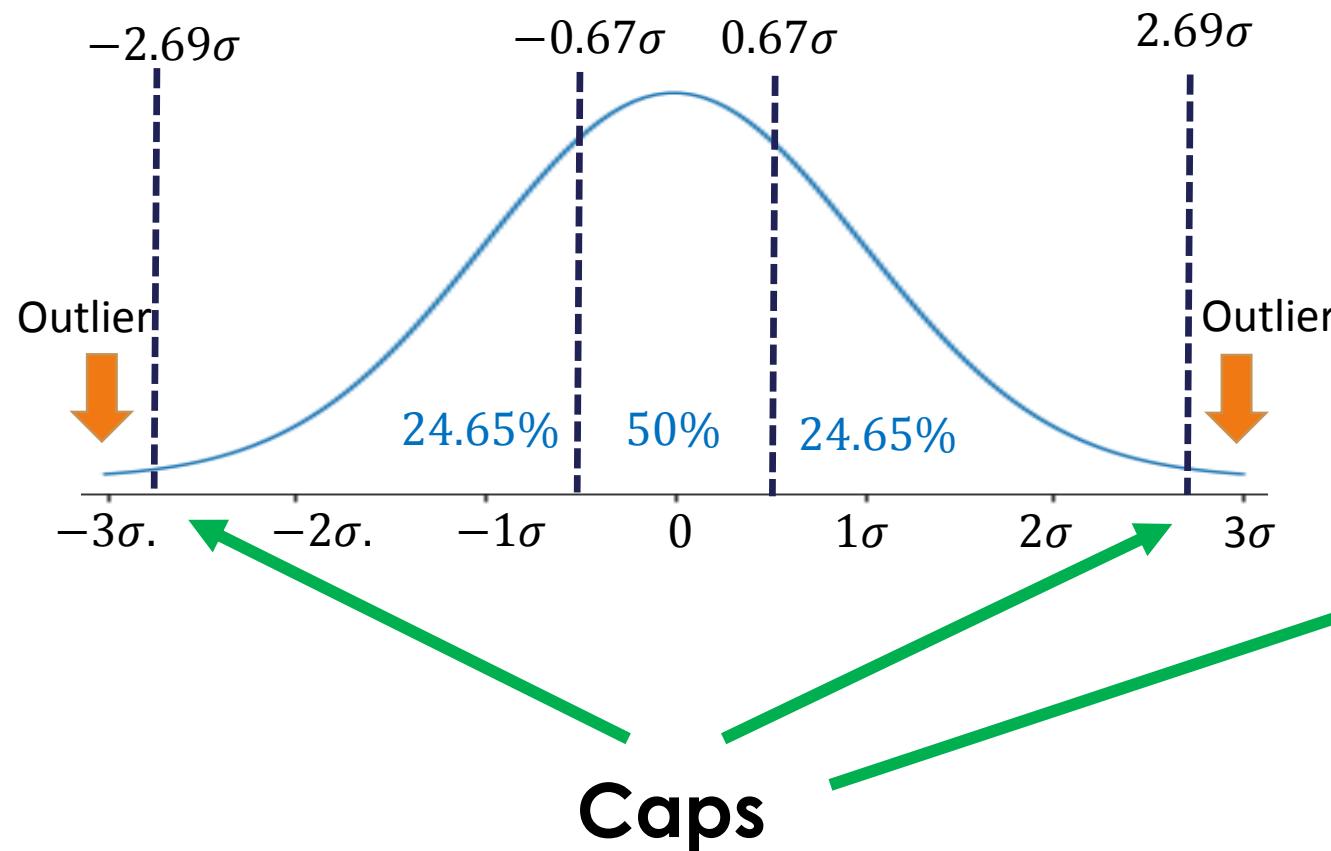
Normal distribution



- ~99% of the observations of a normally distributed variable lie within the mean $\pm 3 \times$ standard deviations.
- Values outside **mean $\pm 3 \times$ standard deviations** are considered outliers



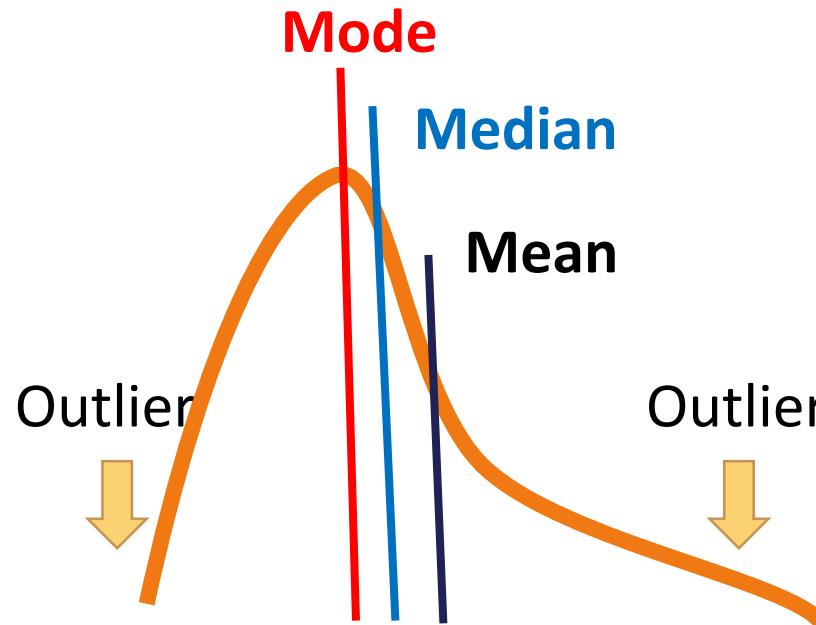
Normal distribution



- ~99% of the observations of a normally distributed variable lie within the mean $\pm 3 \times$ standard deviations.
- Values outside **mean $\pm 3 \times$ standard deviations** are considered outliers



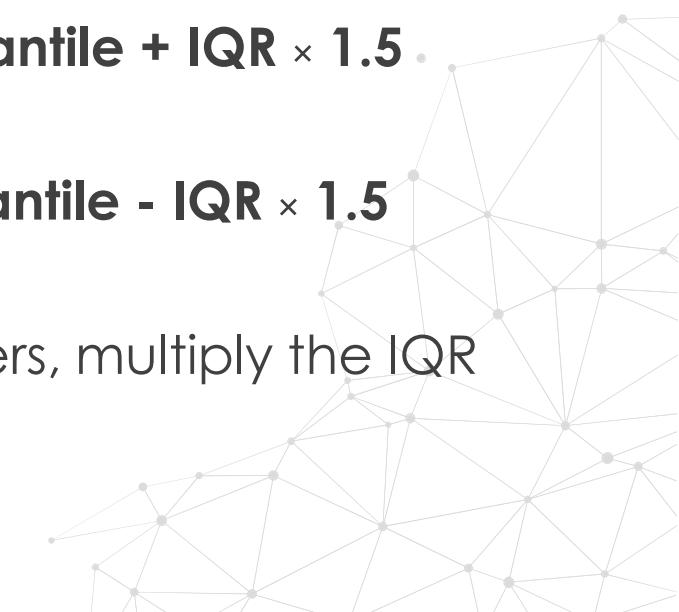
Skewed distributions



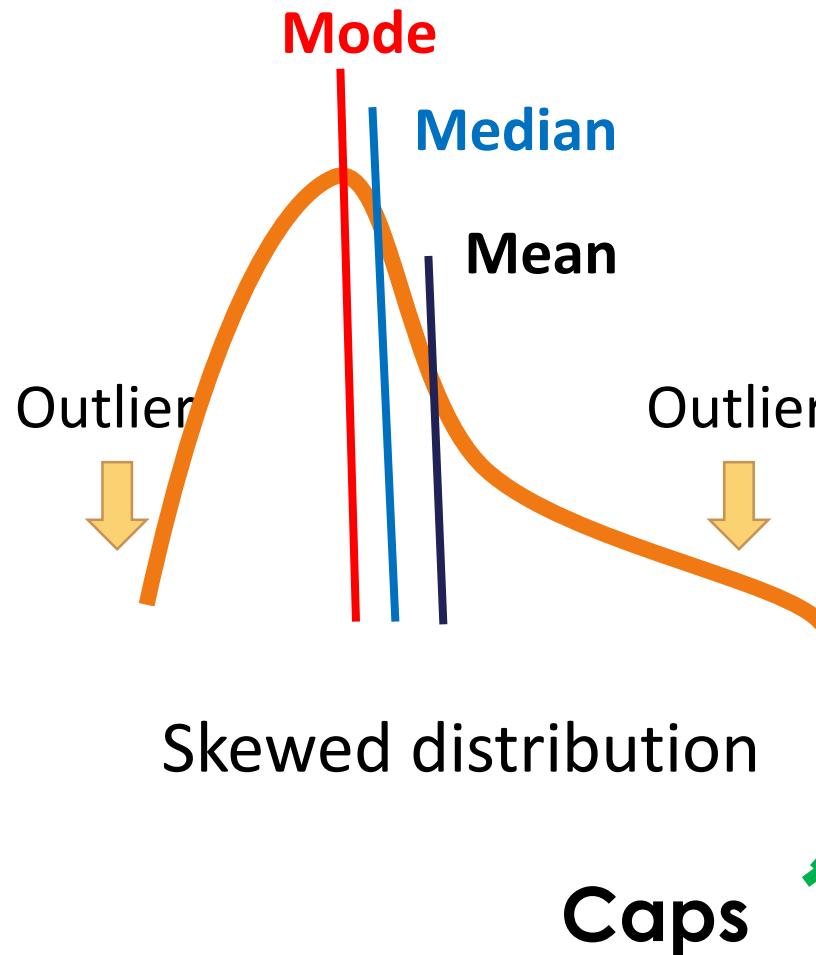
- The general approach is to calculate the quantiles, and then the inter-quantile range (IQR), as follows:

- $IQR = 75^{\text{th}} \text{ Quantile} - 25^{\text{th}} \text{ Quantile}$
- Upper limit = $75^{\text{th}} \text{ Quantile} + IQR \times 1.5$.**
- Lower limit = $25^{\text{th}} \text{ Quantile} - IQR \times 1.5$**

Note, for extreme outliers, multiply the IQR by 3 instead of 1.5



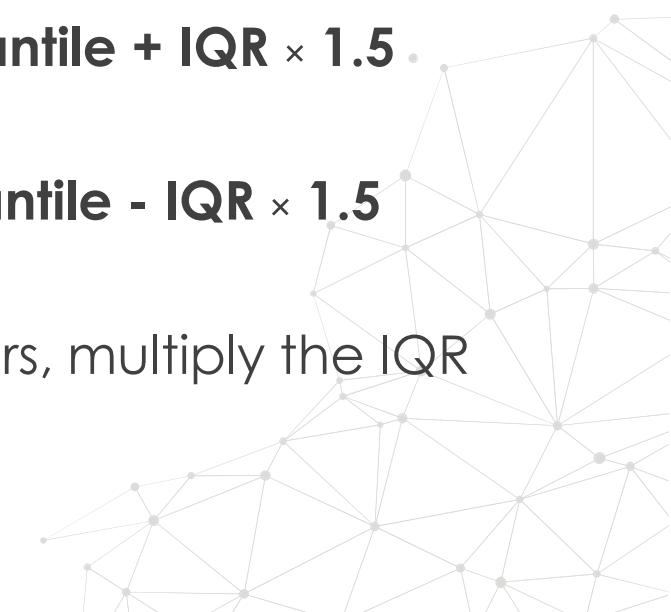
Skewed distributions



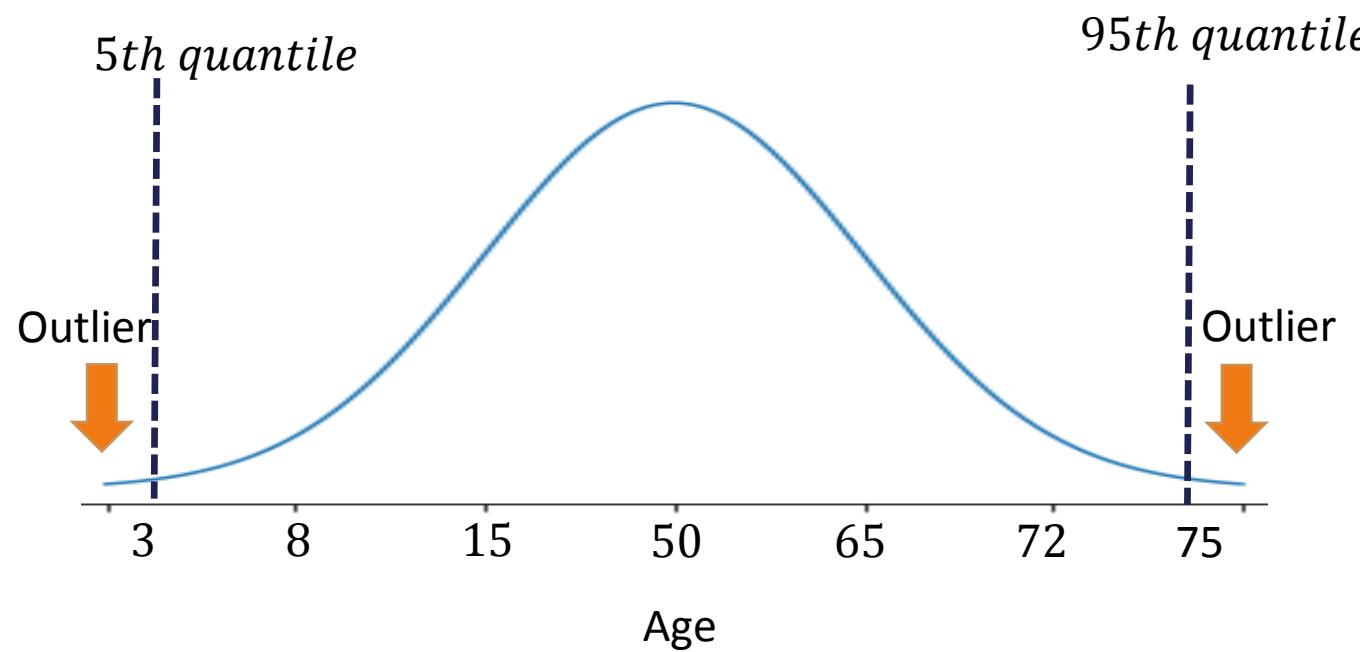
- The general approach is to calculate the quantiles, and then the inter-quantile range (IQR), as follows:

- $IQR = 75^{\text{th}} \text{ Quantile} - 25^{\text{th}} \text{ Quantile}$
- Upper limit = $75^{\text{th}} \text{ Quantile} + IQR \times 1.5$.**
- Lower limit = $25^{\text{th}} \text{ Quantile} - IQR \times 1.5$**

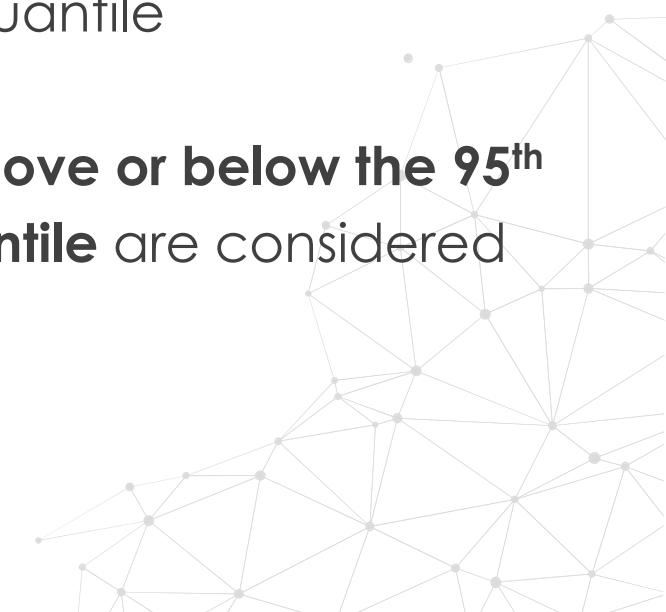
Note, for extreme outliers, multiply the IQR by 3 instead of 1.5



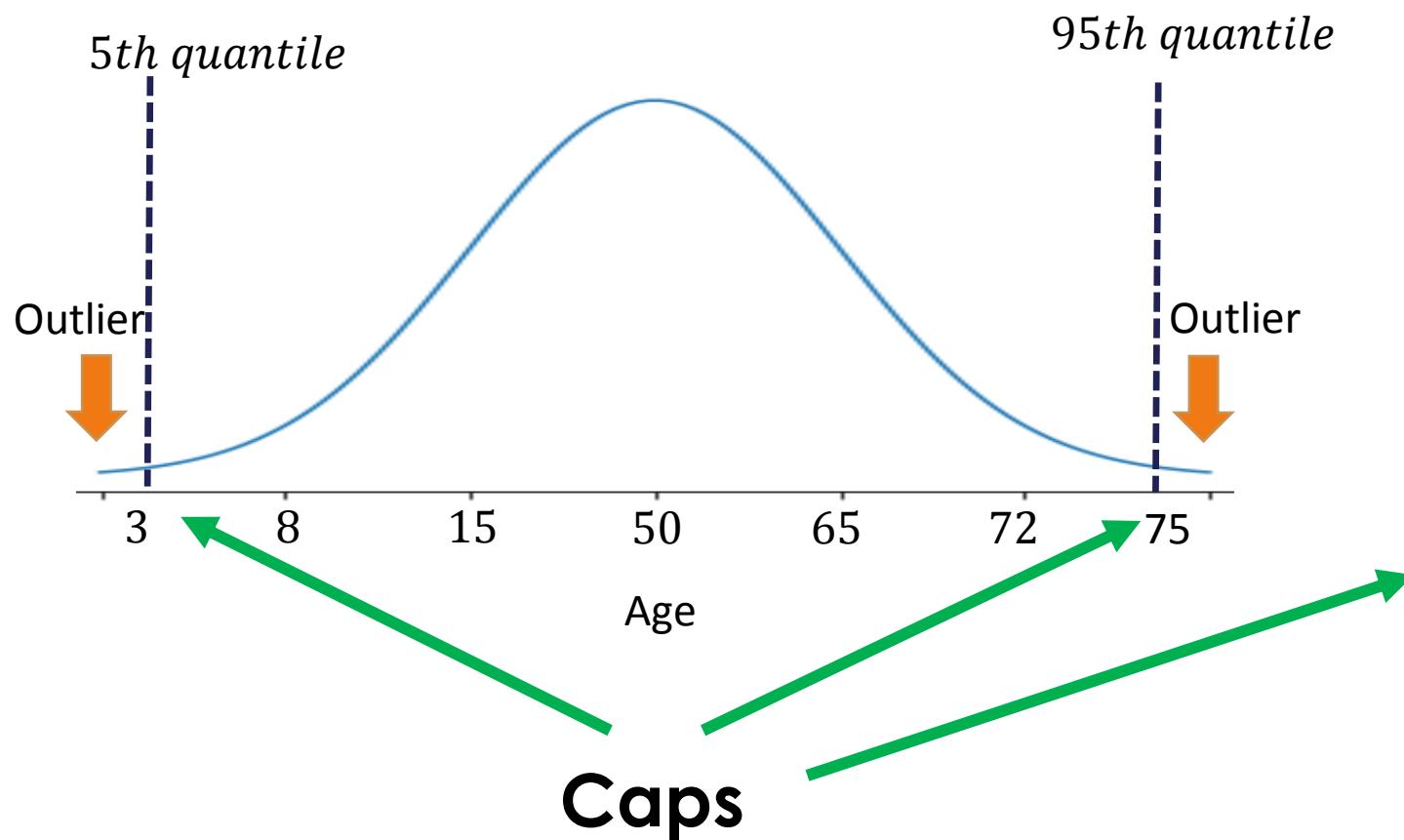
Normal distribution



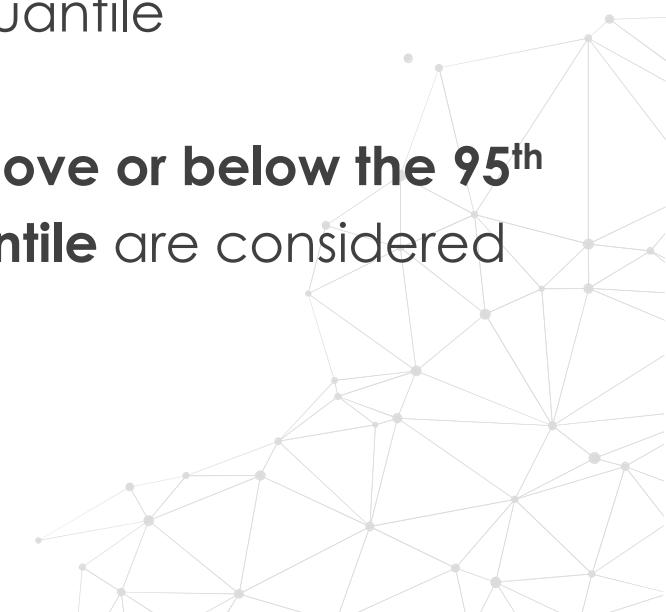
- ~95% of the observations above the 5th quantile
- ~95% of the observations below the 95th quantile
- Values **above or below the 95th or 5th quantile** are considered outliers



Normal distribution



- ~95% of the observations above the 5th quantile
- ~95% of the observations below the 95th quantile
- Values **above or below the 95th or 5th quantile** are considered outliers



• Accompanying Jupyter Notebook



- Accompanying Jupyter Notebook
- How to perform outlier engineering:
 - Gaussian approximation
 - IQR
 - Quantiles
 - Pandas and Feature-engine





THANK YOU

www.trainindata.com





Train In Data

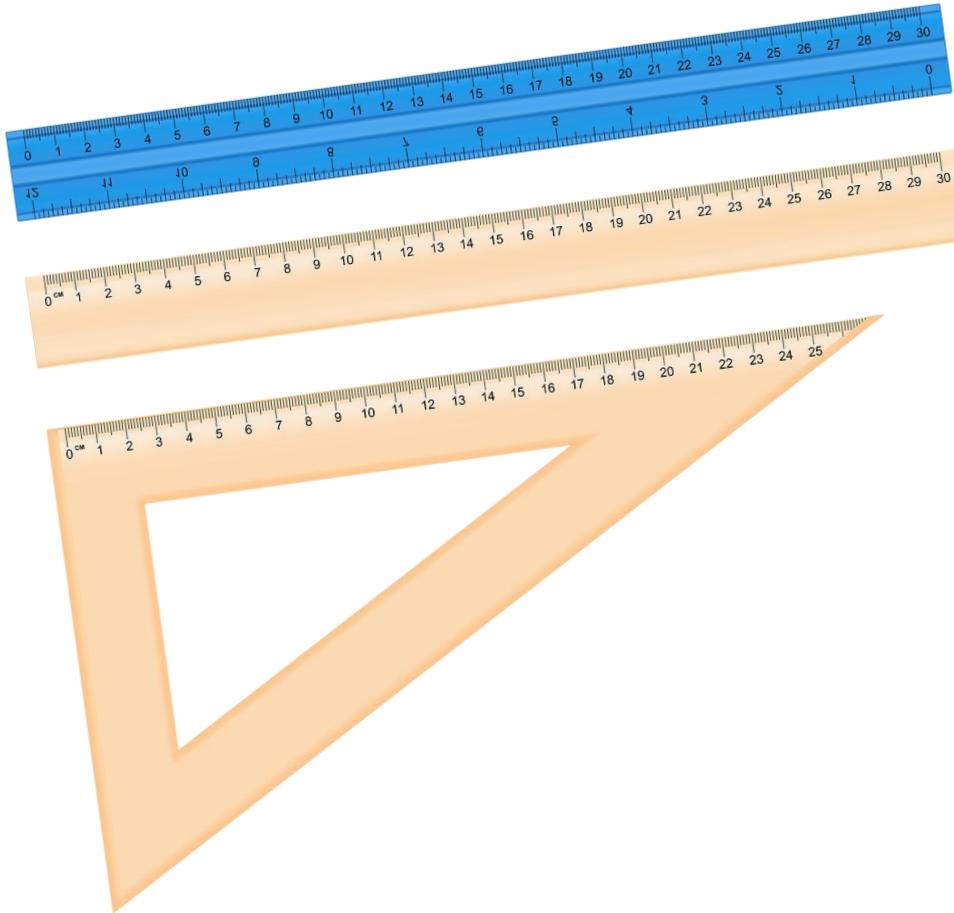
Feature Scaling

• Feature Magnitude matters

- The regression coefficient is directly influenced by the scale of the variable
- Variables with bigger magnitude / value range dominate over the ones with smaller magnitude / value range
- Gradient descent converges faster when features are on similar scales
- Feature scaling helps decrease the time to find support vectors for SVMs
- Euclidean distances are sensitive to feature magnitude.



Algorithms sensitive to magnitude

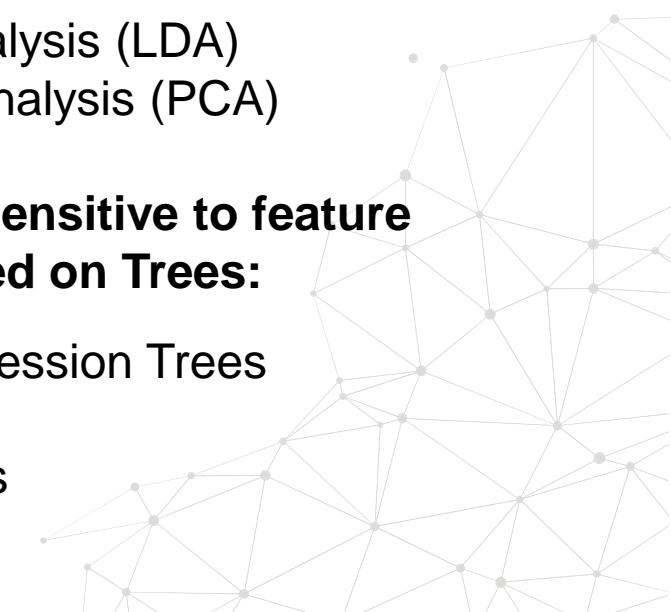


The machine learning models affected by the magnitude of the feature:

- Linear and Logistic Regression
- Neural Networks
- Support Vector Machines
- KNN
- K-means clustering
- Linear Discriminant Analysis (LDA)
- Principal Component Analysis (PCA)

Machine learning models insensitive to feature magnitude are the ones based on Trees:

- Classification and Regression Trees
- Random Forests
- Gradient Boosted Trees

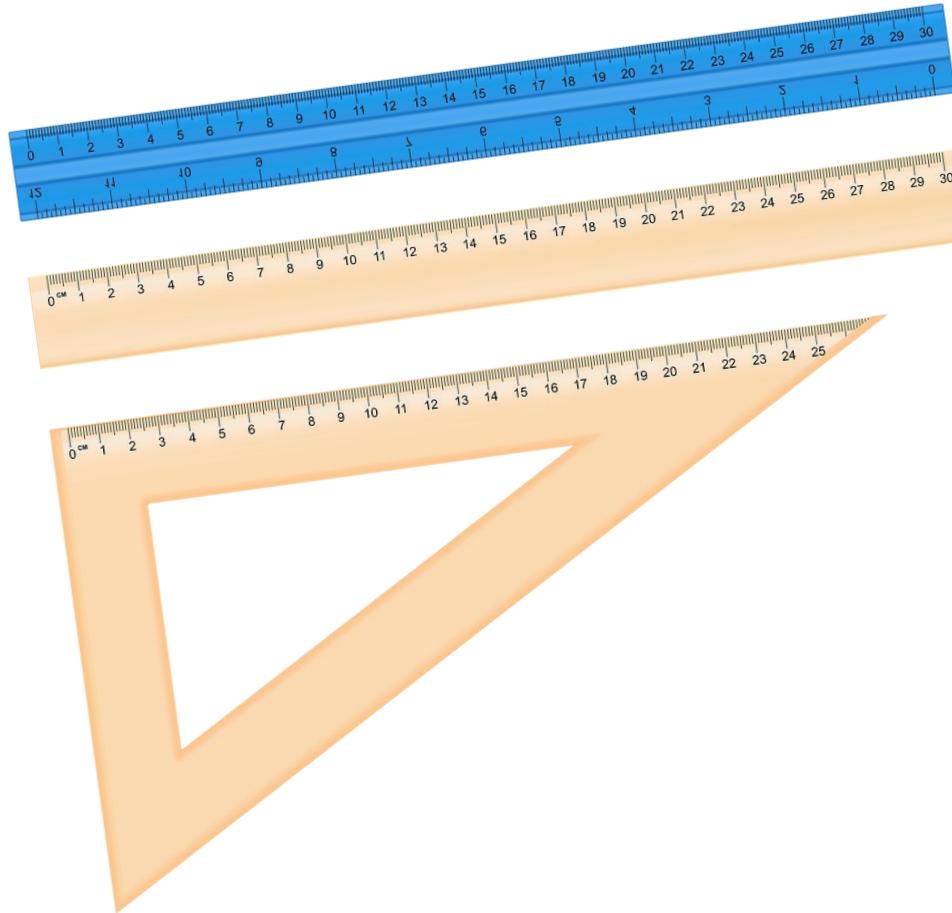


• Feature Scaling

- Feature scaling refers to the methods used to normalize the range of values of independent variables.
- In other words, the methods to set the feature value range within a similar scale.
- Feature scaling is generally the last step in the data pre-processing pipeline, performed just before training the machine learning algorithms.

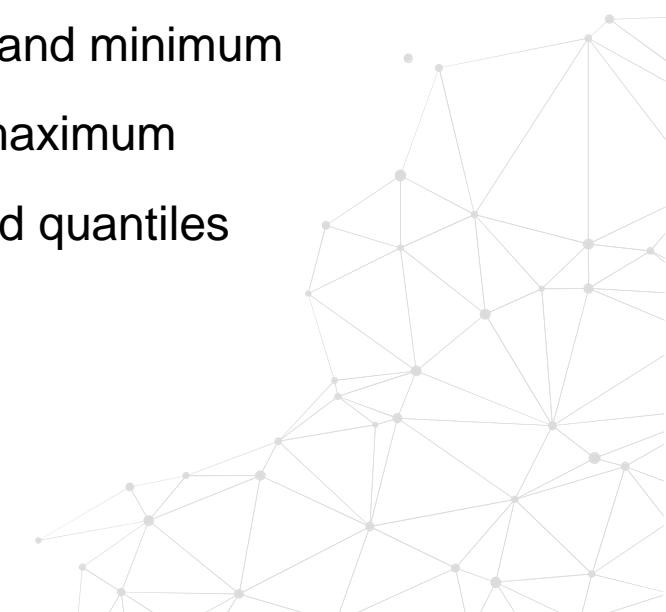


• Feature scaling methods

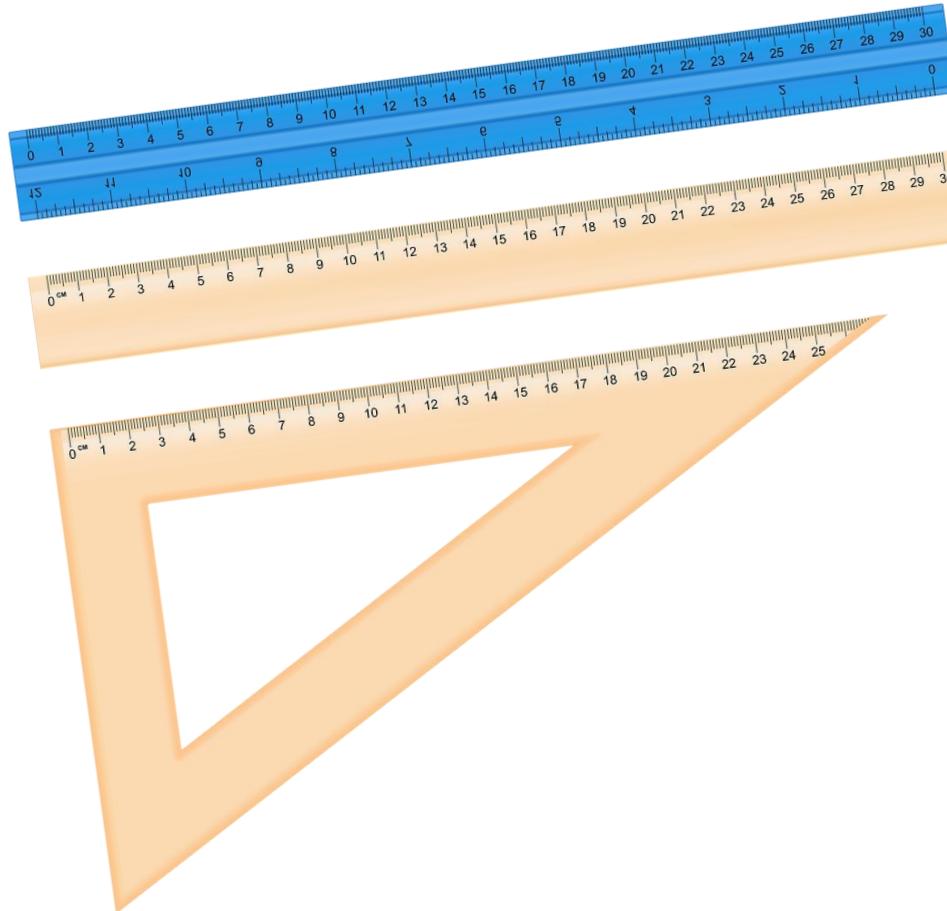


Scaling methods

- Standardisation
- Mean normalisation
- Scaling to maximum and minimum
- Scaling to absolute maximum
- Scaling to median and quantiles
- Scaling to unit norm

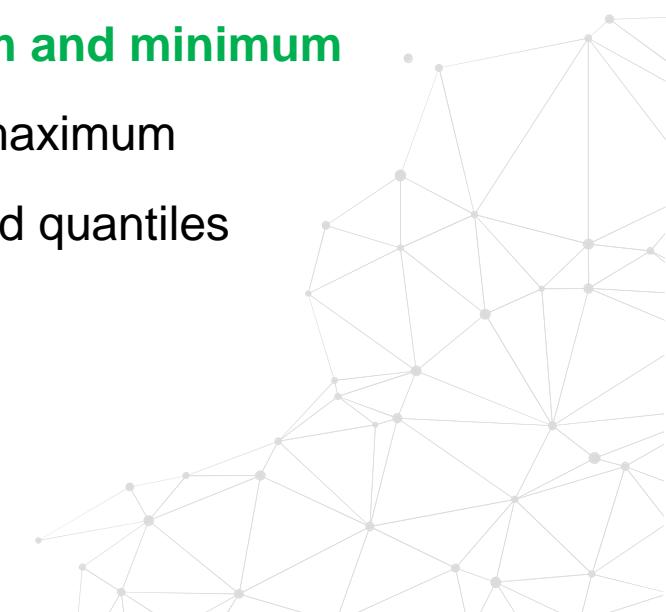


• Feature scaling methods



Scaling methods

- **Standardisation**
- Mean normalisation
- **Scaling to maximum and minimum**
- Scaling to absolute maximum
- Scaling to median and quantiles
- Scaling to unit norm



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
- Demo on how to implement feature selection algorithms using Scikit-learn





THANK YOU

www.trainindata.com





Train In Data

Standardisation

• Standardisation

- Centres the variable at zero and sets the variance to 1.

$$\text{Z-score} = \frac{X - \text{mean}(X)}{\text{std}(X)}$$



Standardisation: example

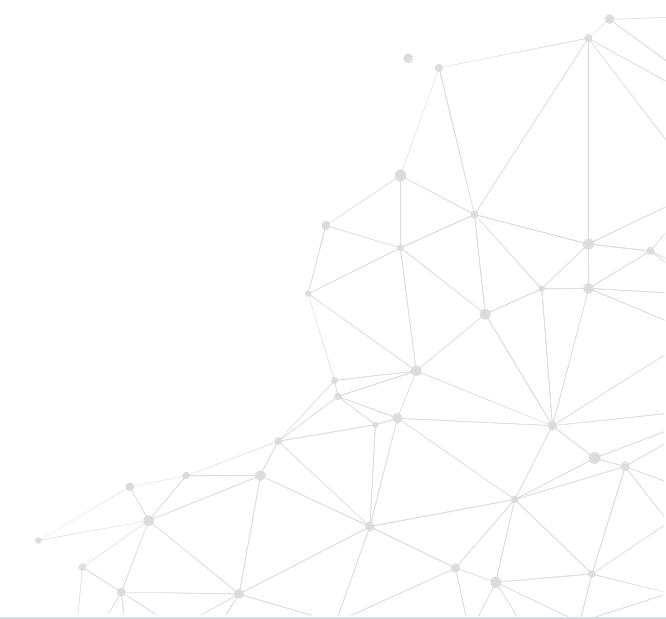
Price
100
90
50
40
20
100
50
60
120
40
200

Mean = 79
Standard dev = 51

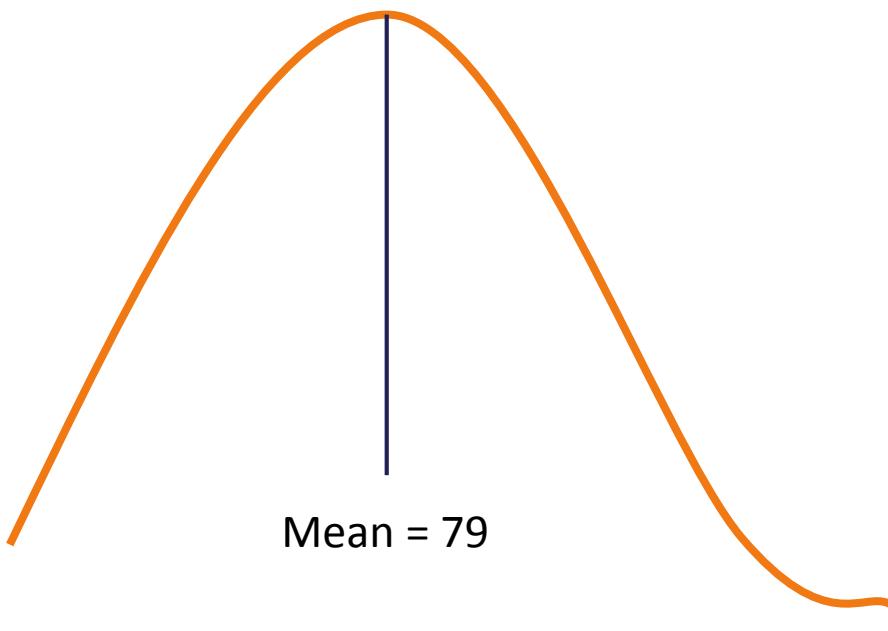


Price
0.41
0.22
-0.57
-0.76
-1.16
0.41
-0.57
-0.37
0.80
-0.76
2.37

$$\frac{\text{Obs.} - \text{Mean}}{\text{Standard dev}}$$



Standardisation: effect

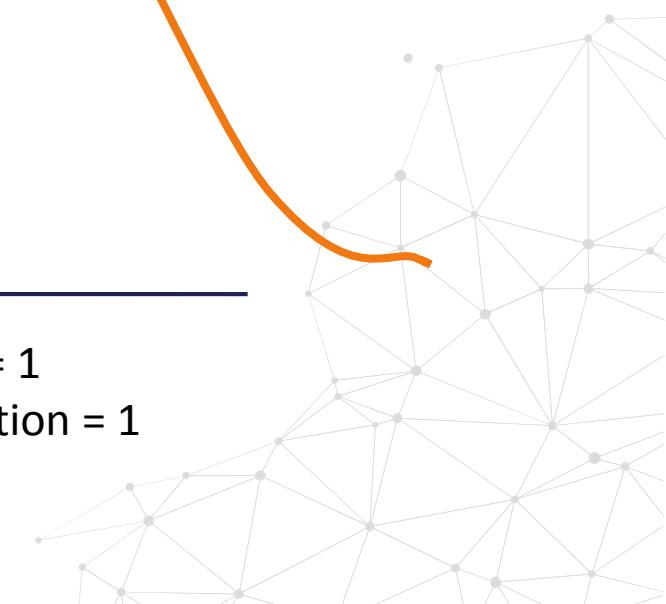
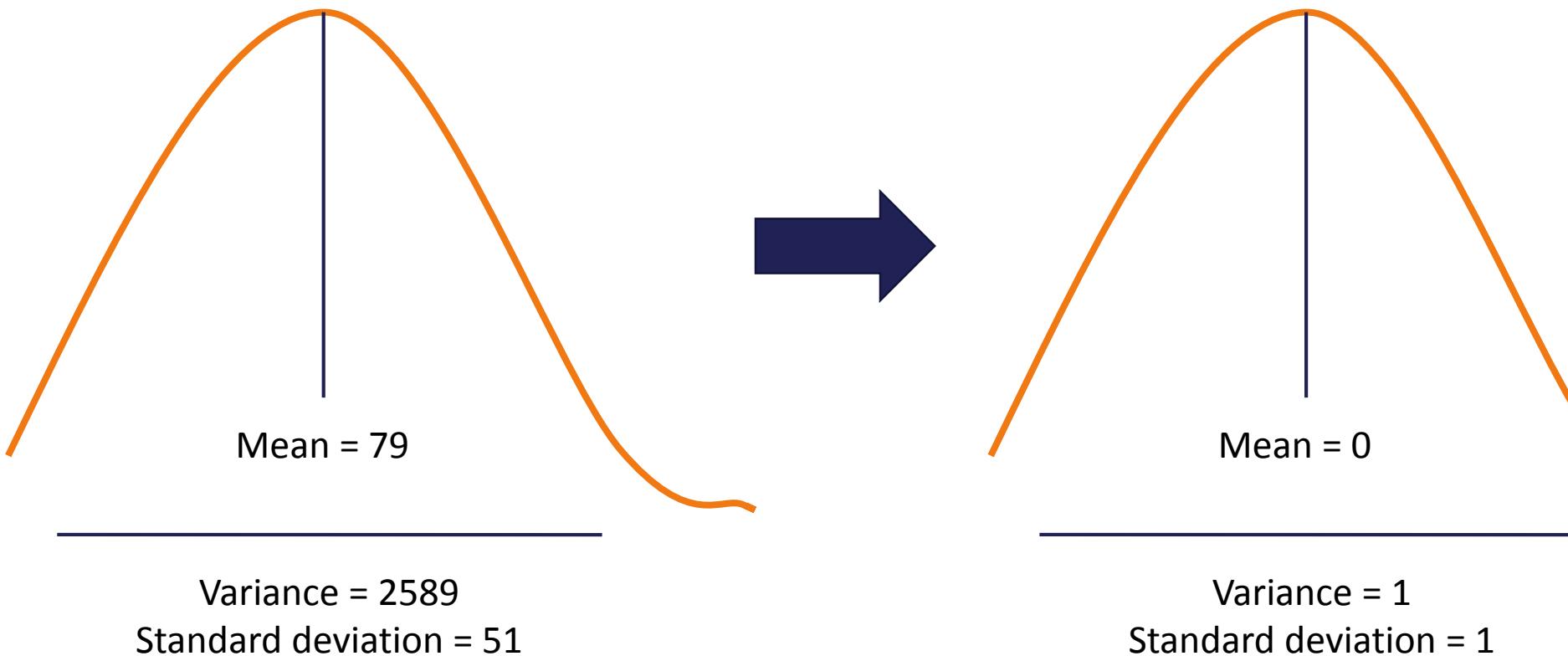


Mean = 79

Variance = 2589
Standard deviation = 51

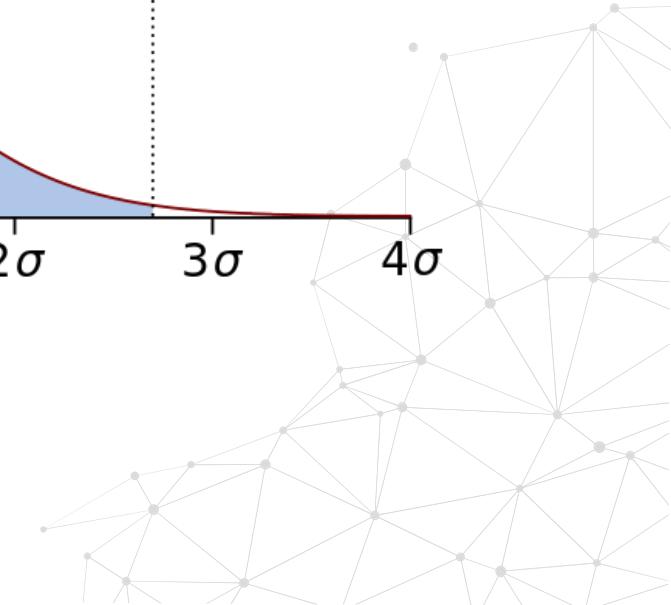
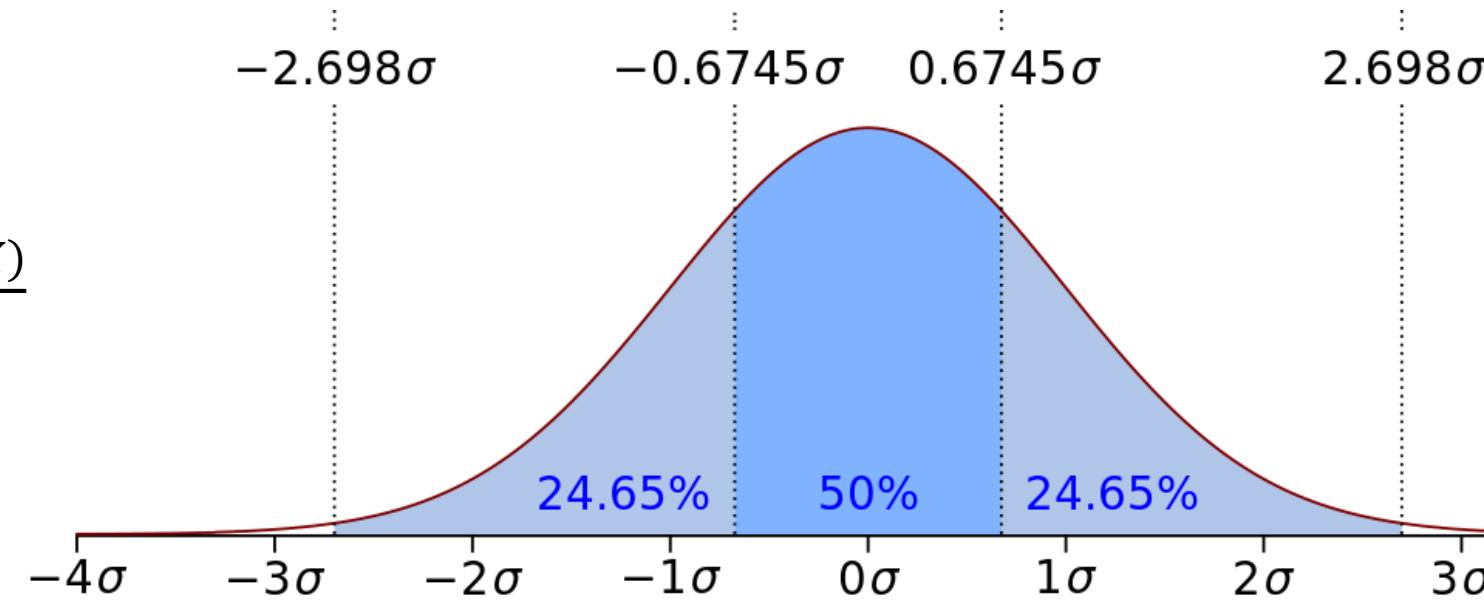


Standardisation: effect



Standardised variable: meaning

$$Z\text{-score} = \frac{X - \text{mean}(X)}{\text{Std}(X)}$$

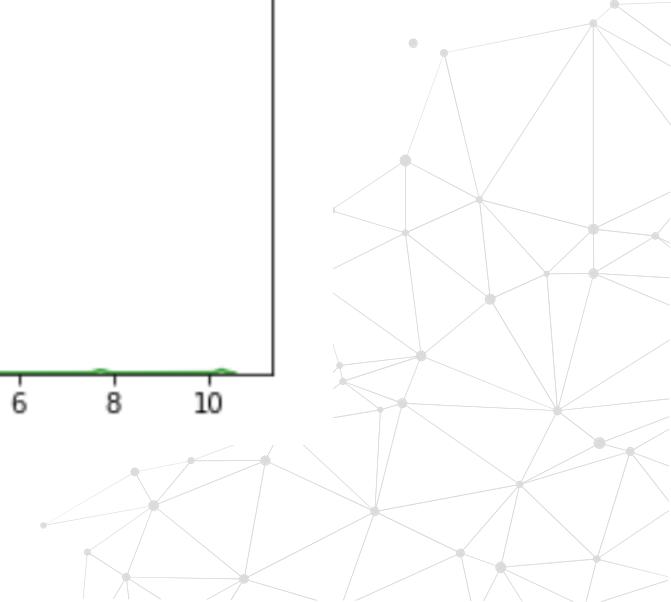
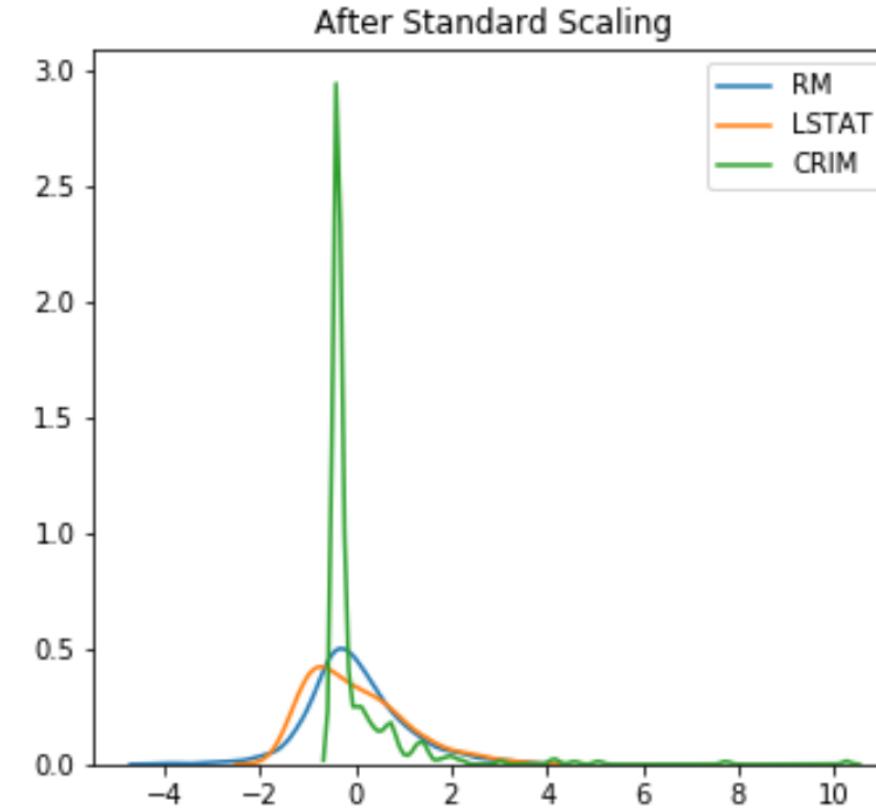
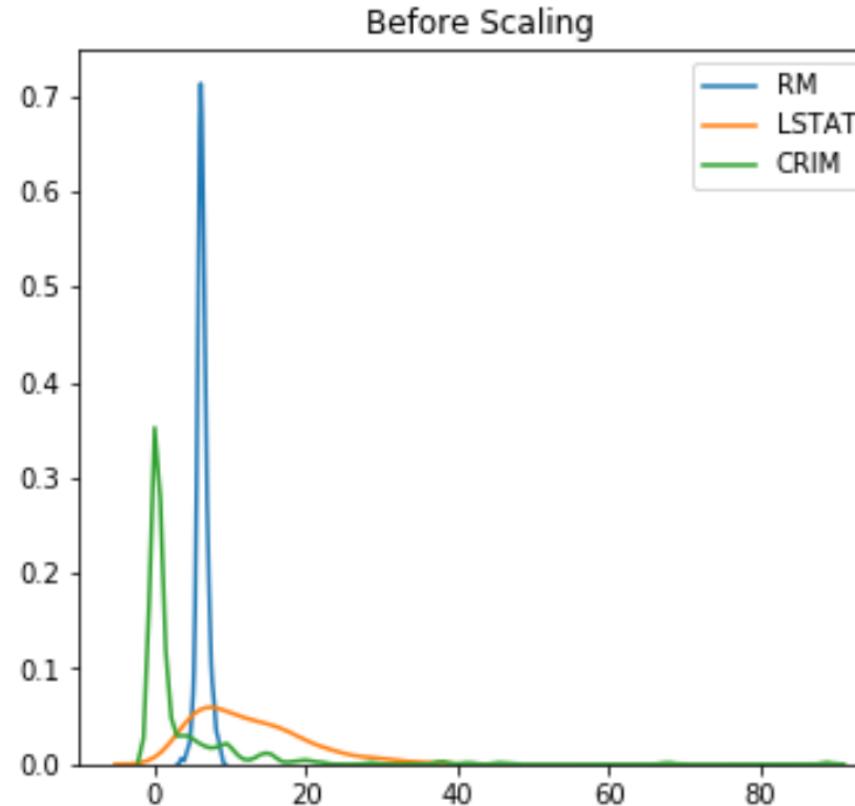


• Standardisation: summary

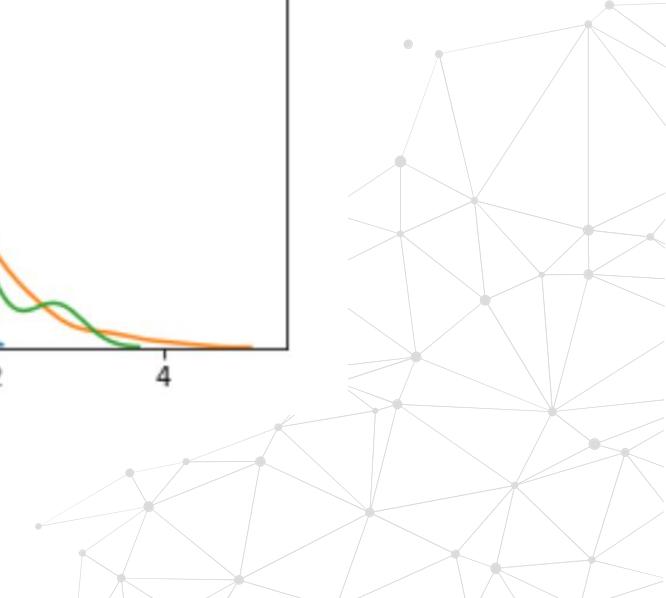
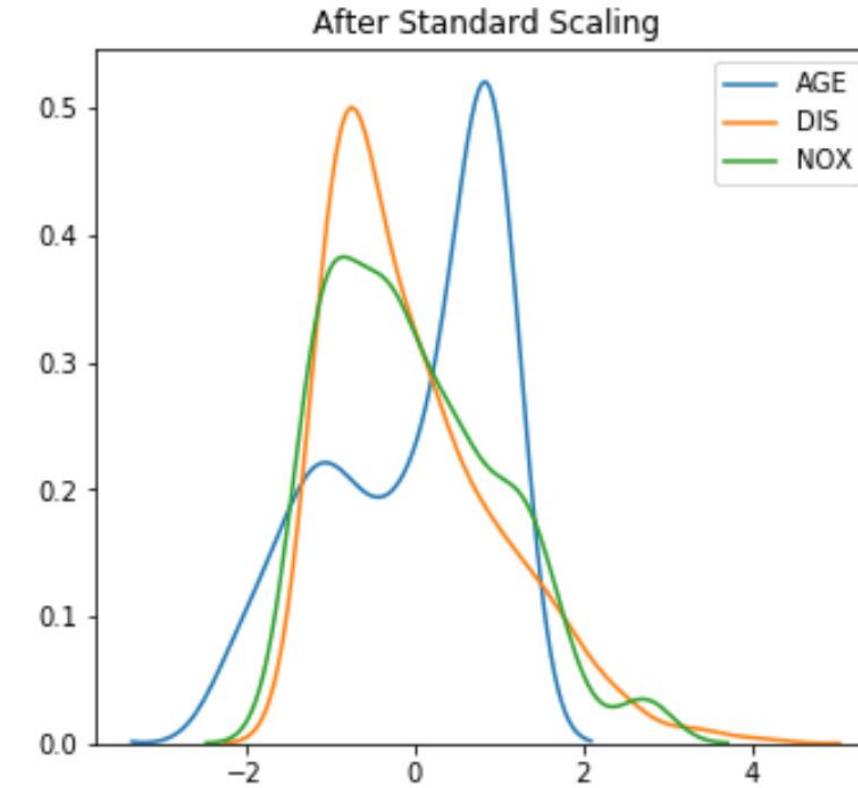
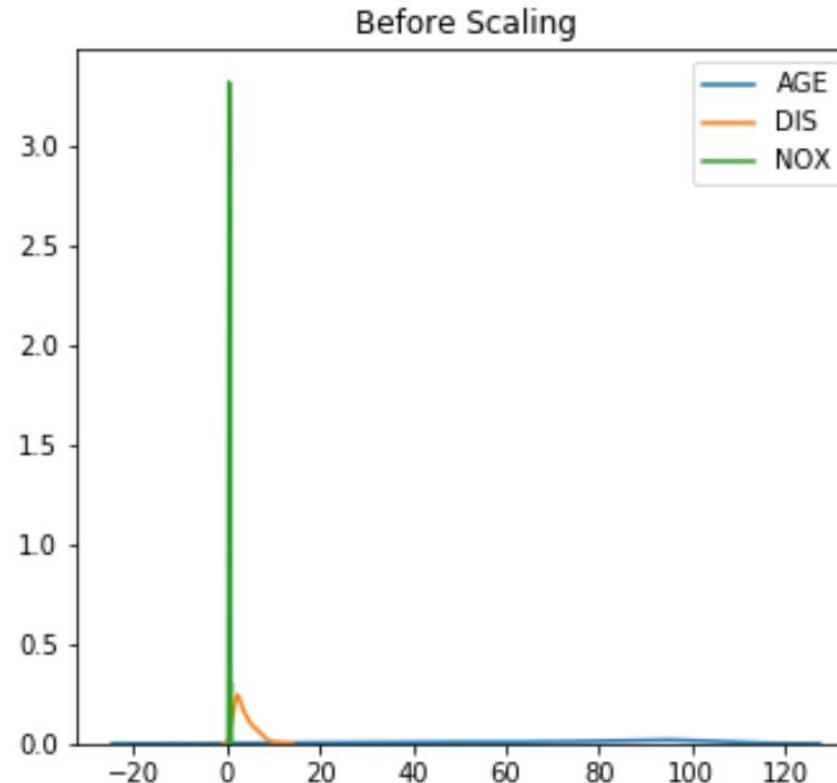
- Centres the mean at 0
- Scales the variance at 1
- Preserves the shape of the original distribution
- Minimum and maximum values vary
- Preserves outliers



Standardisation: Notebook examples



• Standardisation: Notebook examples



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
- Standardisation with Scikit-learn





THANK YOU

www.trainindata.com





Mean Normalisation

• Mean Normalisation

- Centres the variable at 0 and re-scales the variable to the value range.

$$x_{\text{scaled}} = \frac{x - \text{mean}(X)}{\text{max}(X) - \text{min}(X)}$$



Mean normalisation: example

Price
100
90
50
40
20
100
50
60
120
40
200

Mean = 79

Max = 200

Min = 20

Range = $200 - 20 = 180$



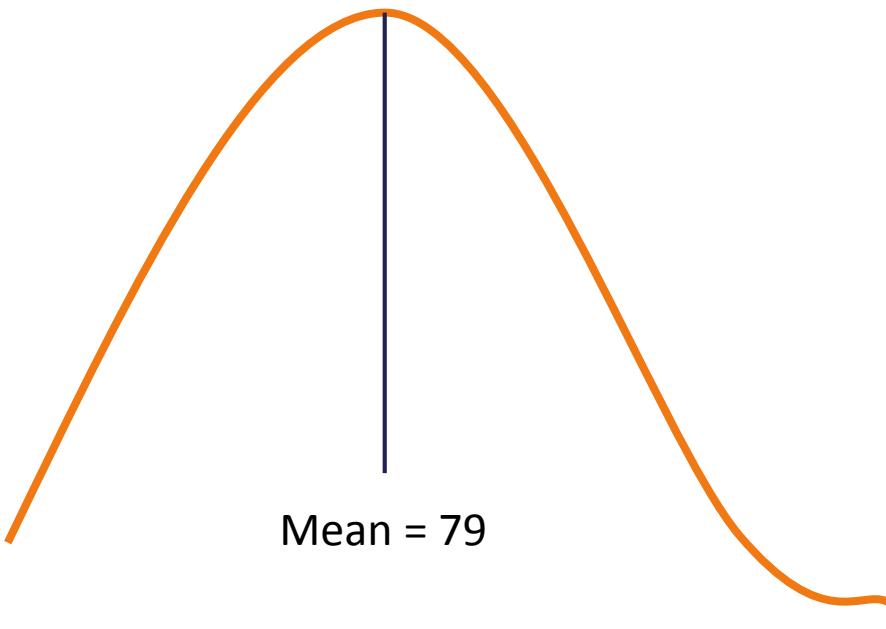
Obs. - Mean

Range

Price
0.12
0.06
-0.16
-0.22
-0.33
0.12
-0.16
-0.11
0.23
-0.22
0.67



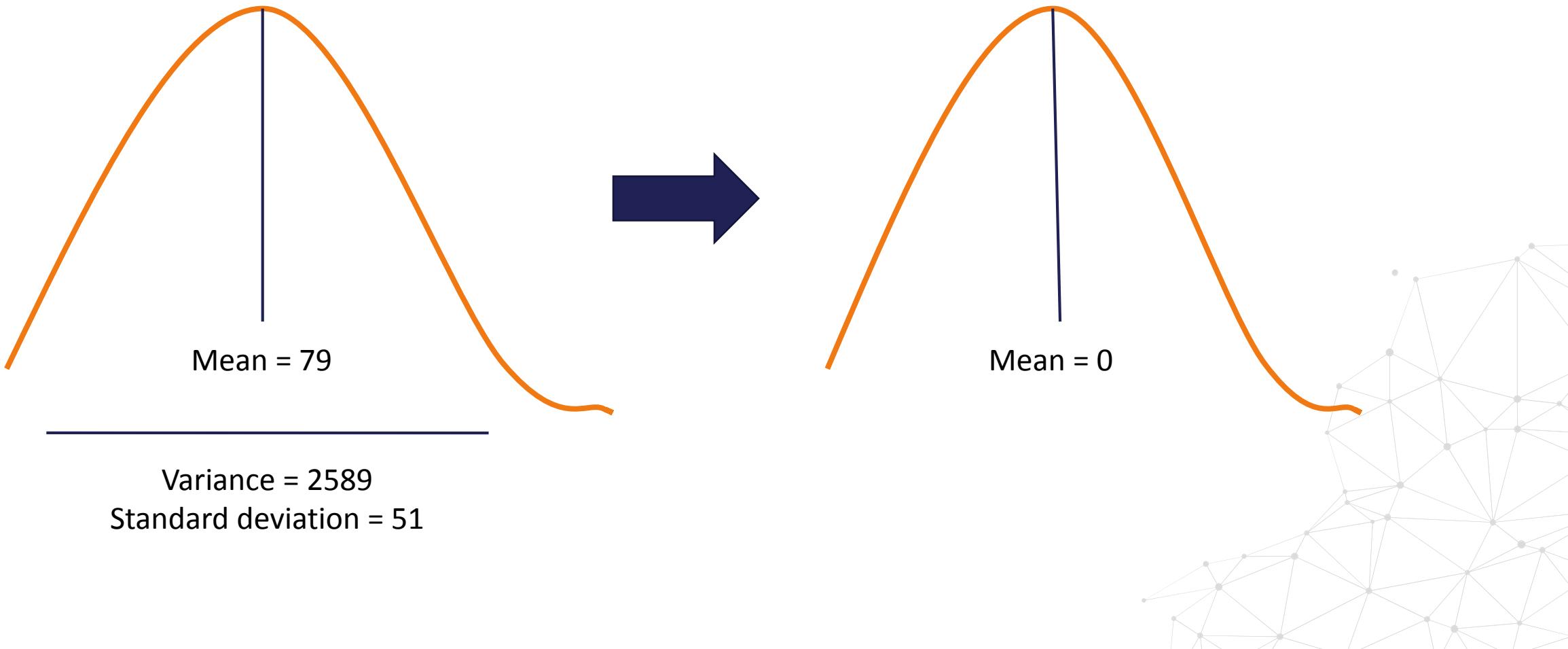
Mean normalisation: effect



Variance = 2589
Standard deviation = 51



Mean normalisation: effect

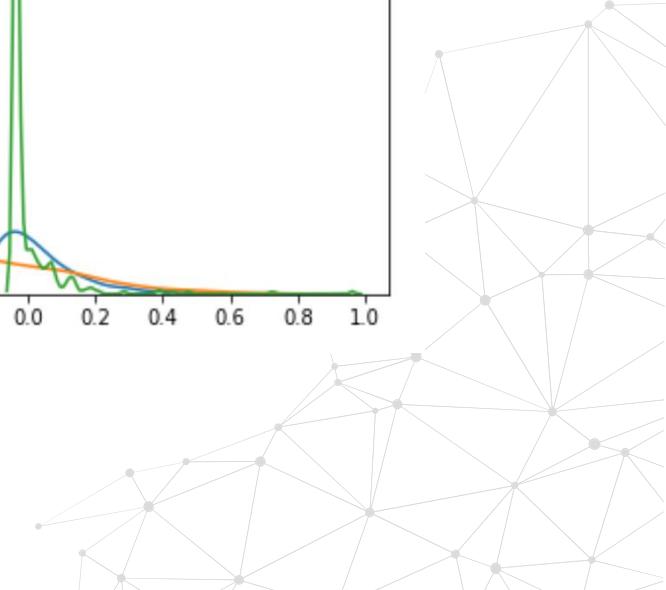
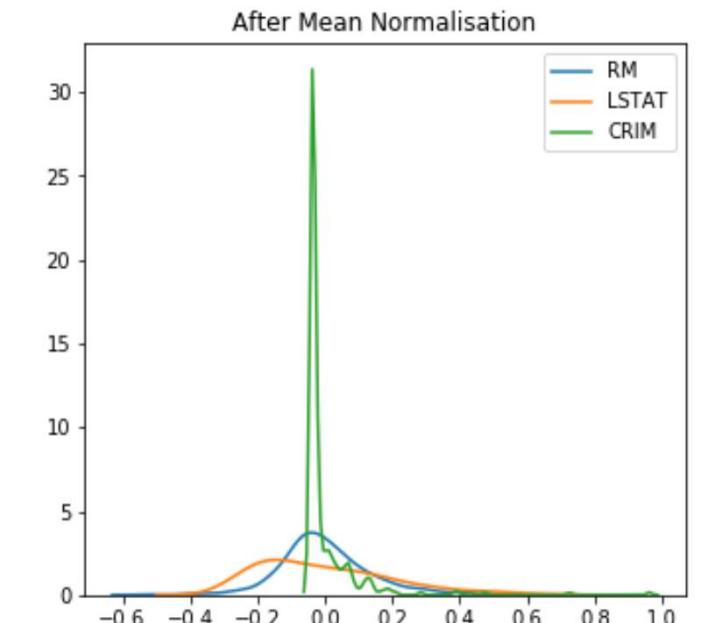
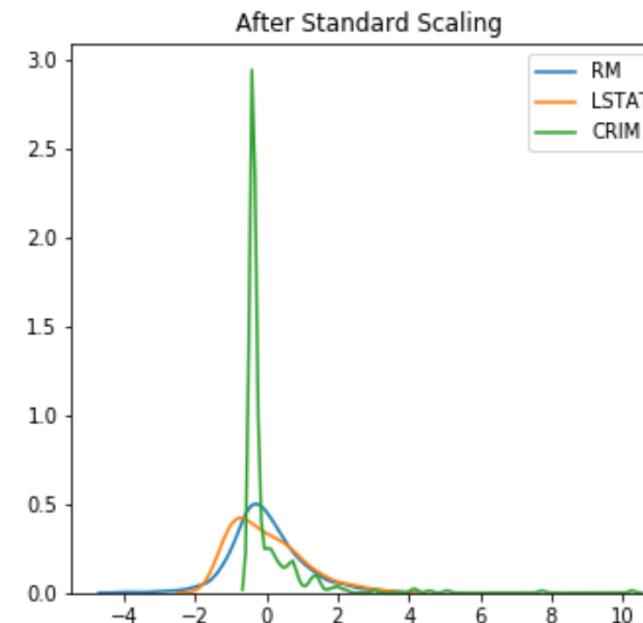
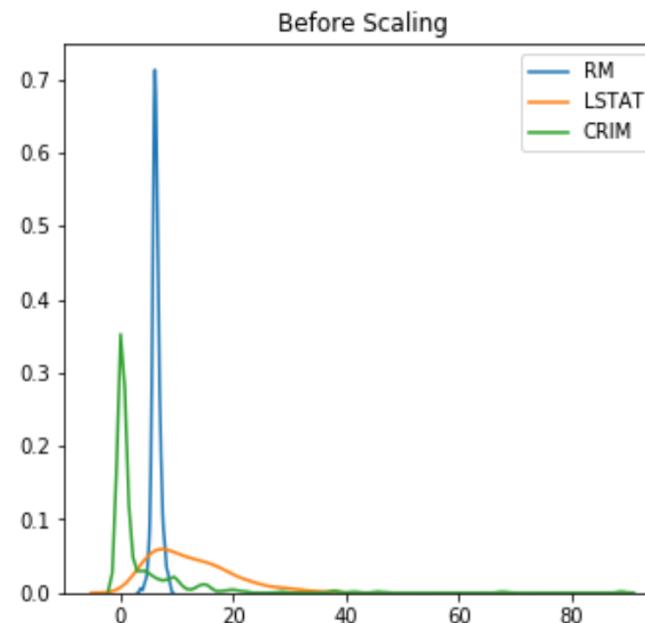


Mean normalisation: summary

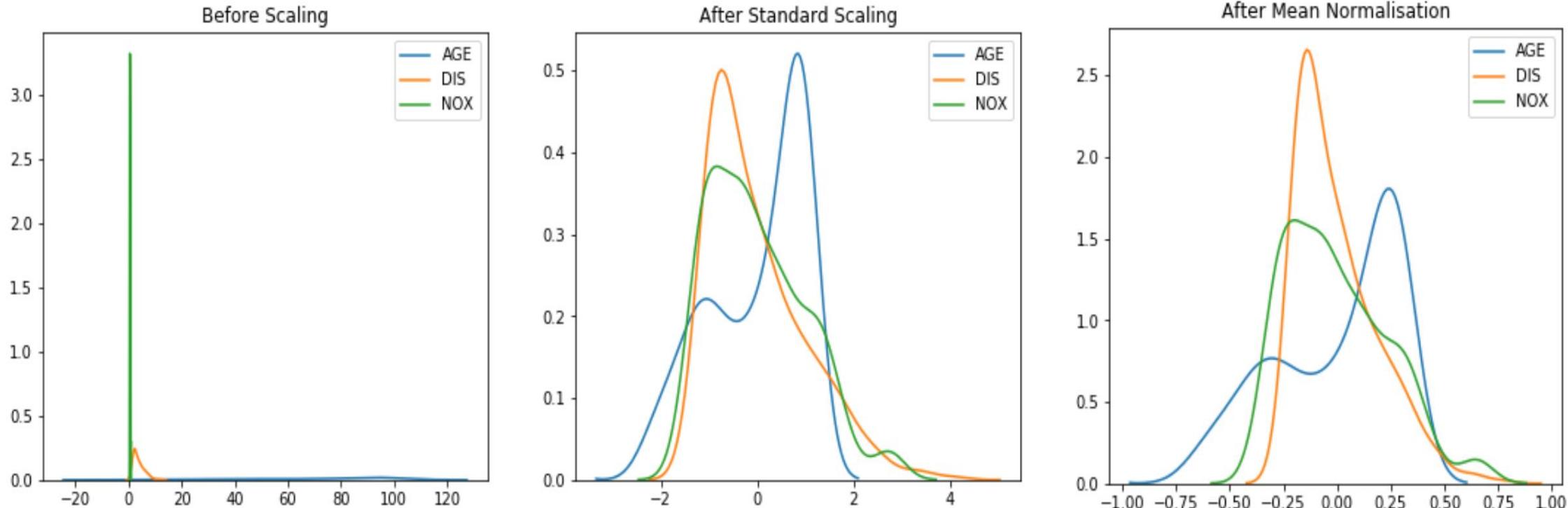
- Centres the mean at 0
- Variance varies
- May alter shape of the original distribution
- Minimum and maximum values within [-1;1]
- Preserves outliers



Mean normalisation: Notebook



Mean normalisation: Notebook



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
- Mean normalisation with pandas and a work-around with Scikit-learn

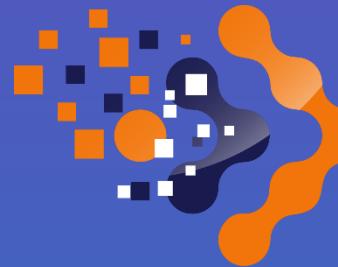




THANK YOU

www.trainindata.com





Train In Data

Min-Max Scaling

• MinMaxScaling

- Scales the variable between 0 and 1

$$x_{\text{scaled}} = \frac{x - \min(X)}{\max(X) - \min(X)}$$



• MinMaxScaling: example

Price
100
90
50
40
20
100
50
60
120
40
200

Max = 200
Min = 20
Range = $200 - 20 = 180$



Obs. - Min

Range

Price
0.44
0.39
0.17
0.11
0.00
0.44
0.17
0.22
0.56
0.11
1.00

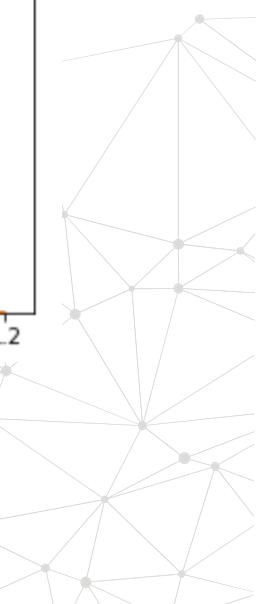
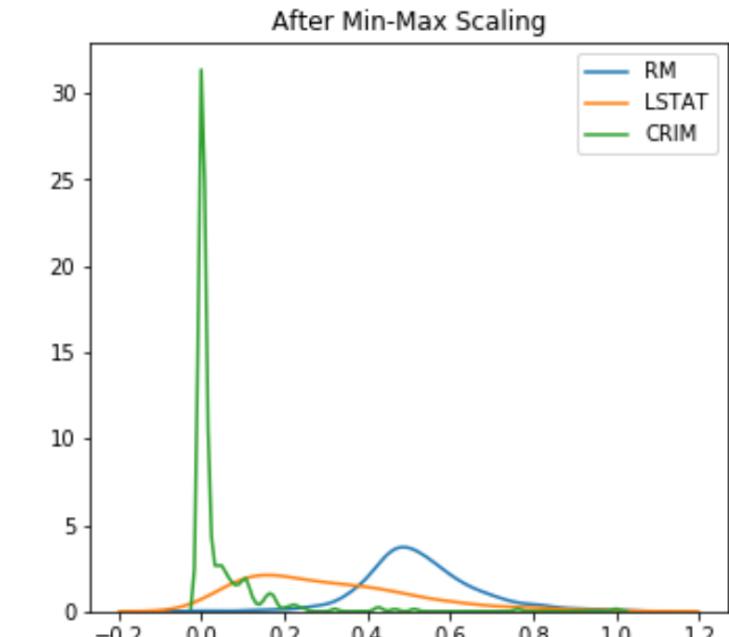
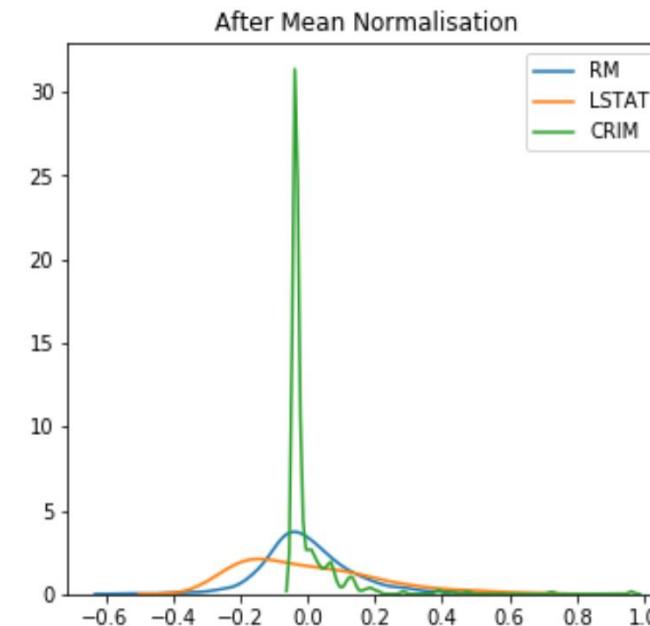
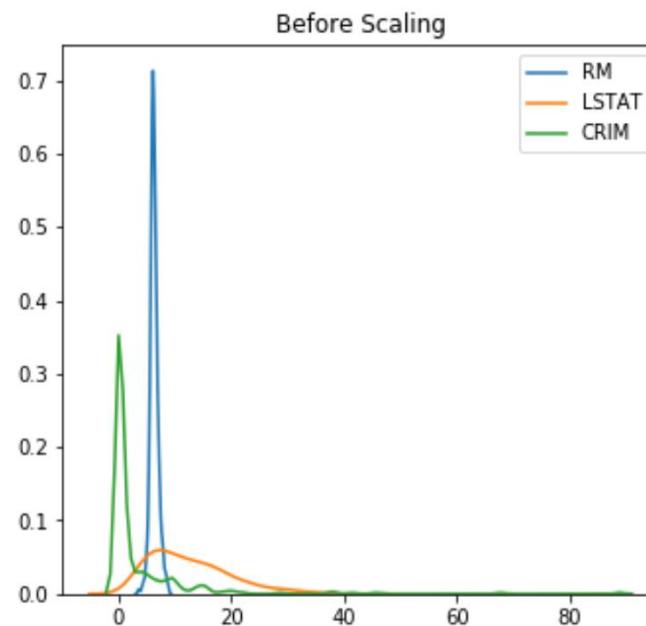


• MinMaxScaling: summary

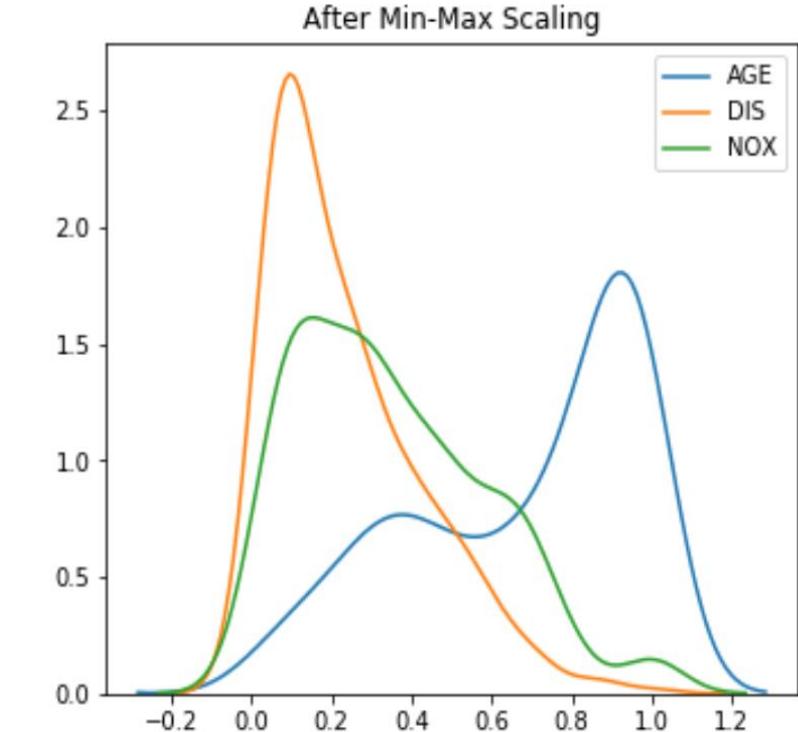
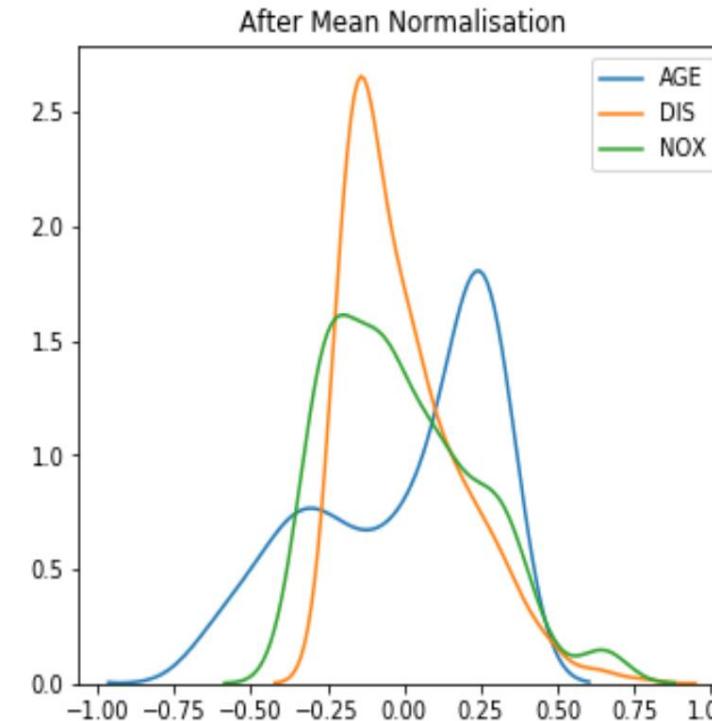
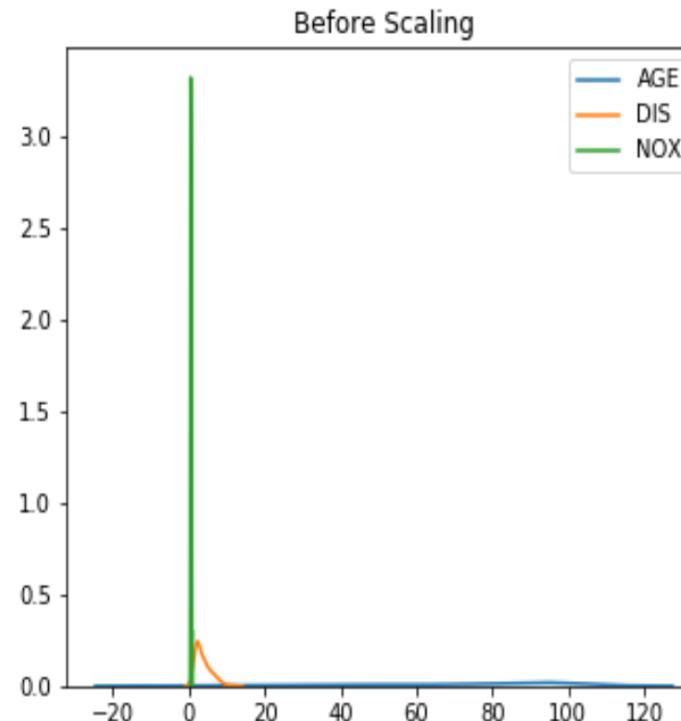
- Mean varies
- Variance varies
- May alter shape of the original distribution
- Minimum and maximum values within $[0;1]$
- Preserves outliers



MinMaxScaling: Notebook



• MinMaxScaling : Notebook



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
- MinMaxScaling with Scikit-learn





THANK YOU

www.trainindata.com





Maximum Absolute Scaling

• MaxAbsScaling

- Scales the variable between -1 and 1

$$x_{\text{scaled}} = \frac{x}{\max(|X|)}$$



MaxAbsScaling: example

Price
100
90
50
40
20
100
50
60
120
40
200

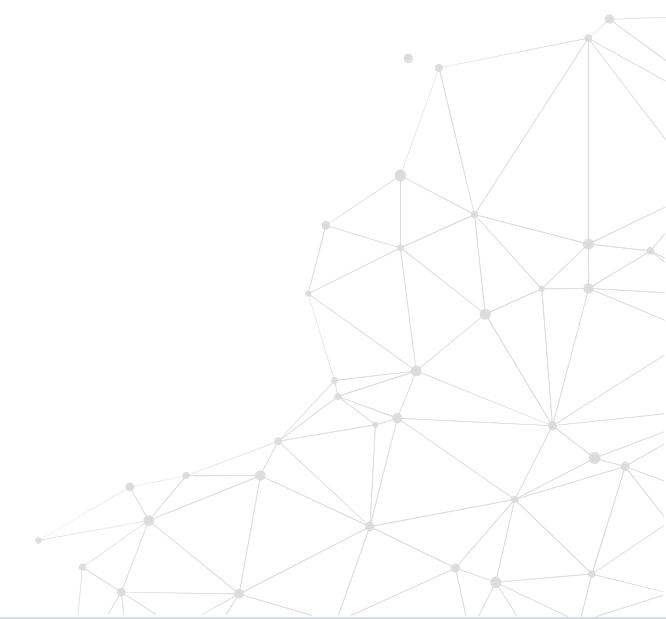
Max = 200



Obs.

Max value

Price
0.50
0.45
0.25
0.20
0.10
0.50
0.25
0.30
0.60
0.20
1.00

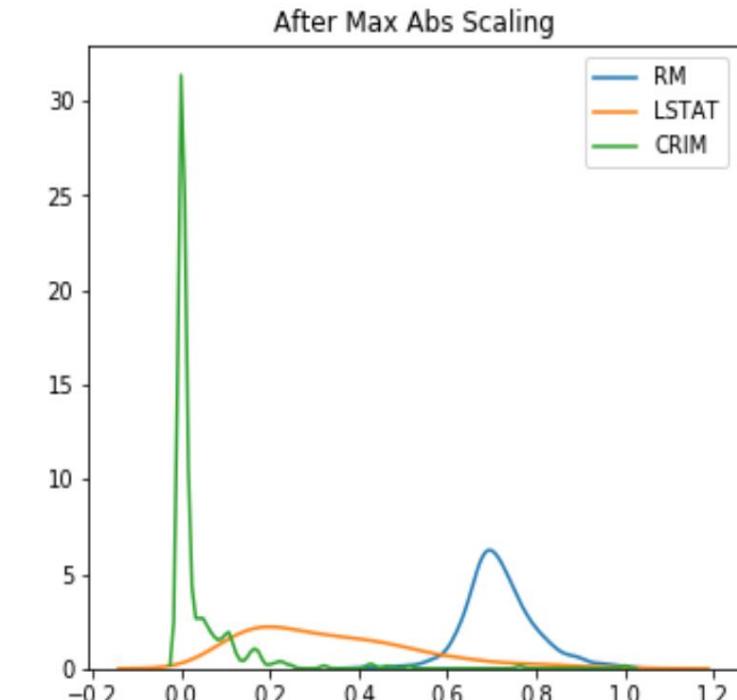
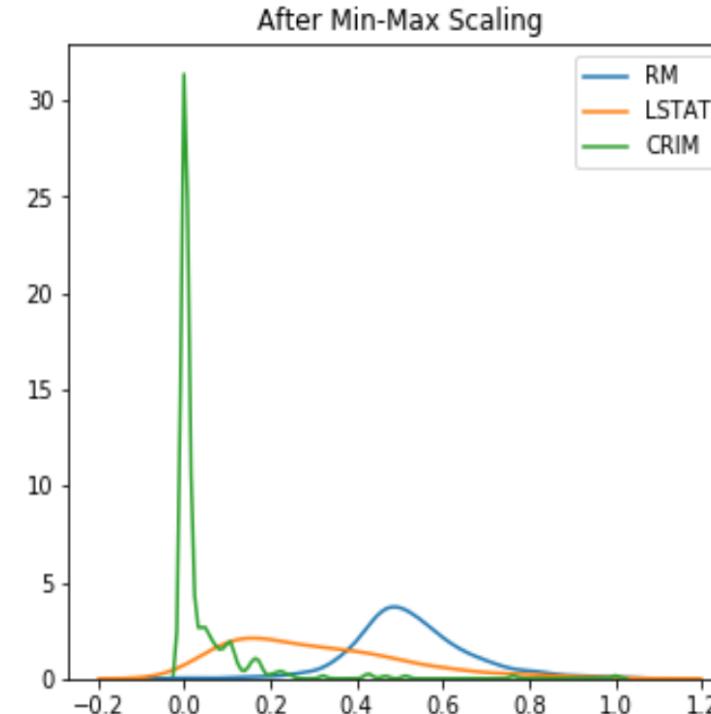
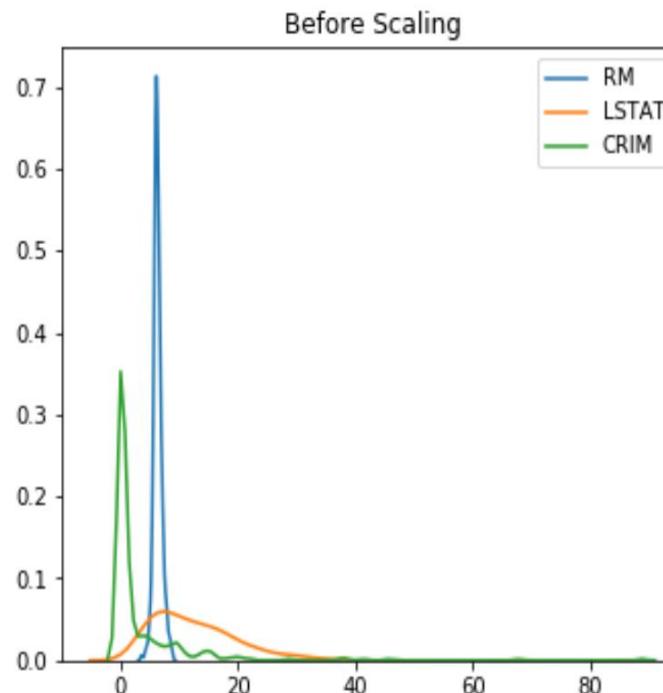


• MaxAbsScaling: summary

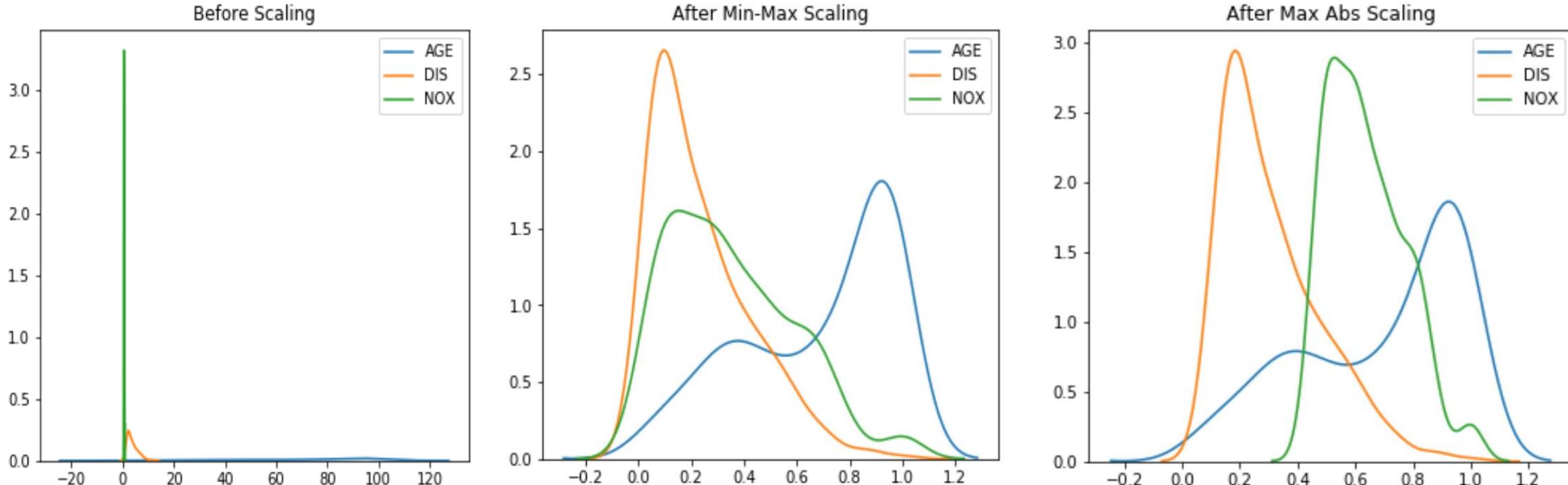
- Mean not centred
- Variance not scaled
- Scikit-learn recommends use with:
 - data that is centred
 - sparse matrices



MaxAbsScaling: Notebook



MaxAbsScaling : Notebook



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
- MaxAbsScaling with Scikit-learn





THANK YOU

www.trainindata.com





Maximum Absolute Scaling

• MaxAbsScaling

- Scales the variable between -1 and 1

$$x_{\text{scaled}} = \frac{x}{\max(|x|)}$$



MaxAbsScaling: example

Price
100
90
50
40
20
100
50
60
120
40
200

Max = 200



Obs.

Max value

Price
0.50
0.45
0.25
0.20
0.10
0.50
0.25
0.30
0.60
0.20
1.00

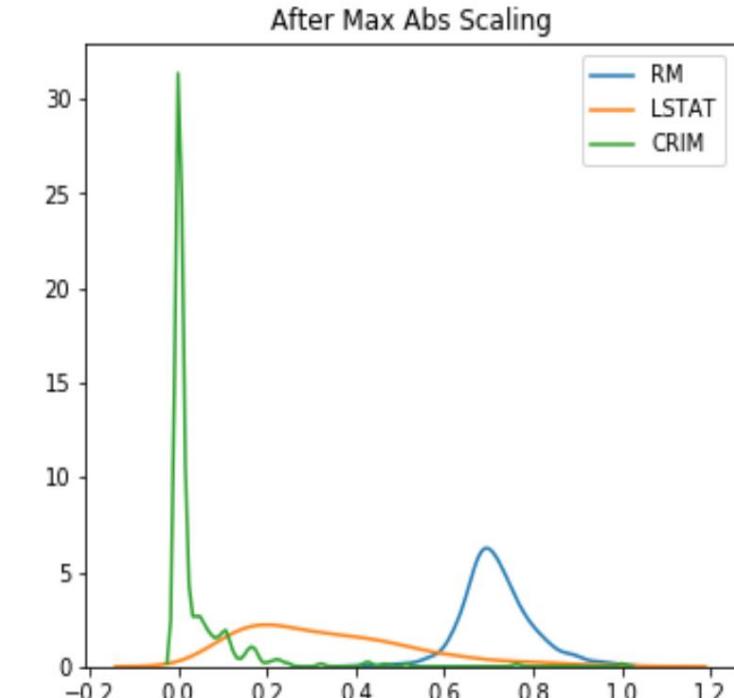
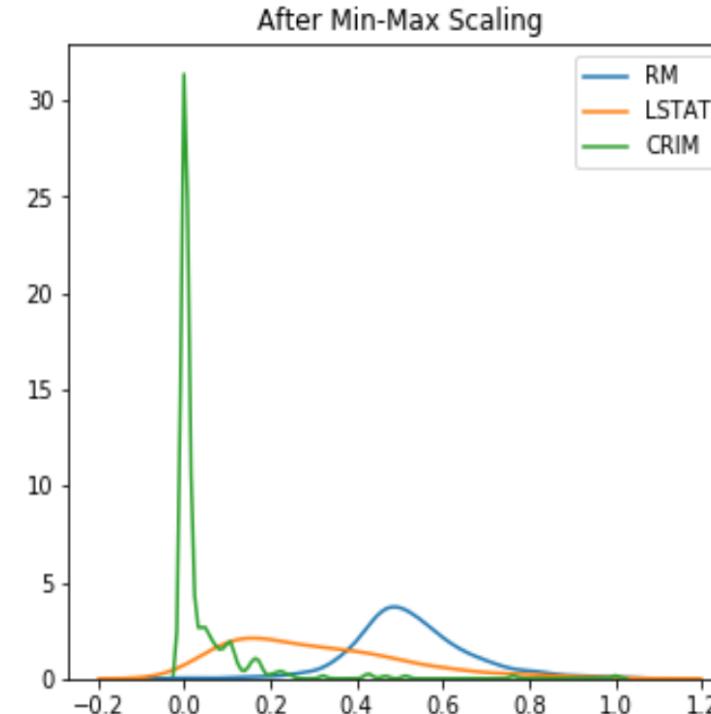
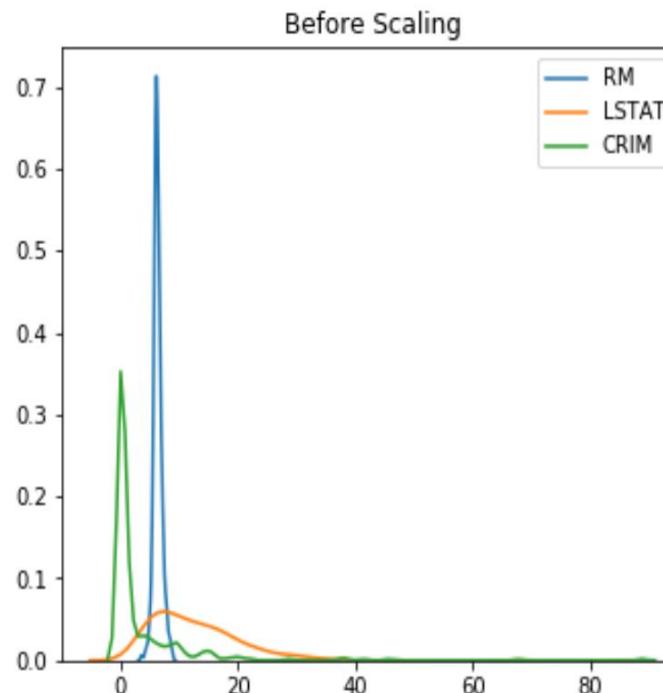


• MaxAbsScaling: summary

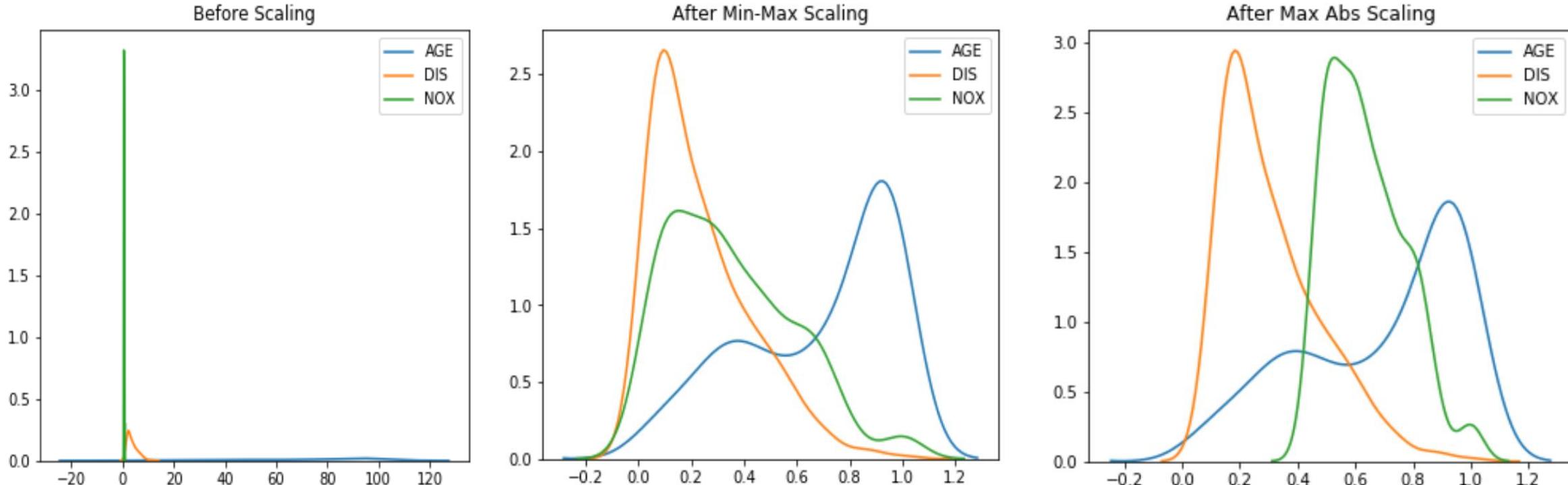
- Mean not centred
- Variance not scaled
- Scikit-learn recommends use with:
 - data that is centred
 - sparse matrices



MaxAbsScaling: Notebook



MaxAbsScaling : Notebook



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
- MaxAbsScaling with Scikit-learn





THANK YOU

www.trainindata.com





Scaling to median and IQR

• Robust Scaling

$$x_{\text{scaled}} = \frac{x - \text{median}(X)}{\text{75th quant}(X) - \text{25th quant}(X)}$$



• Robust Scaling: example

Price
100
90
50
40
20
100
50
60
120
40
200

Median = 60
25th quantile = 45
75th quantile = 100
IQR = 100 – 45 = 55



$$\frac{\text{Obs.} - \text{Median}}{\text{IQR}}$$

Price
0.73
0.55
-0.18
-0.36
-0.73
0.73
-0.18
0.00
1.09
-0.36
2.55

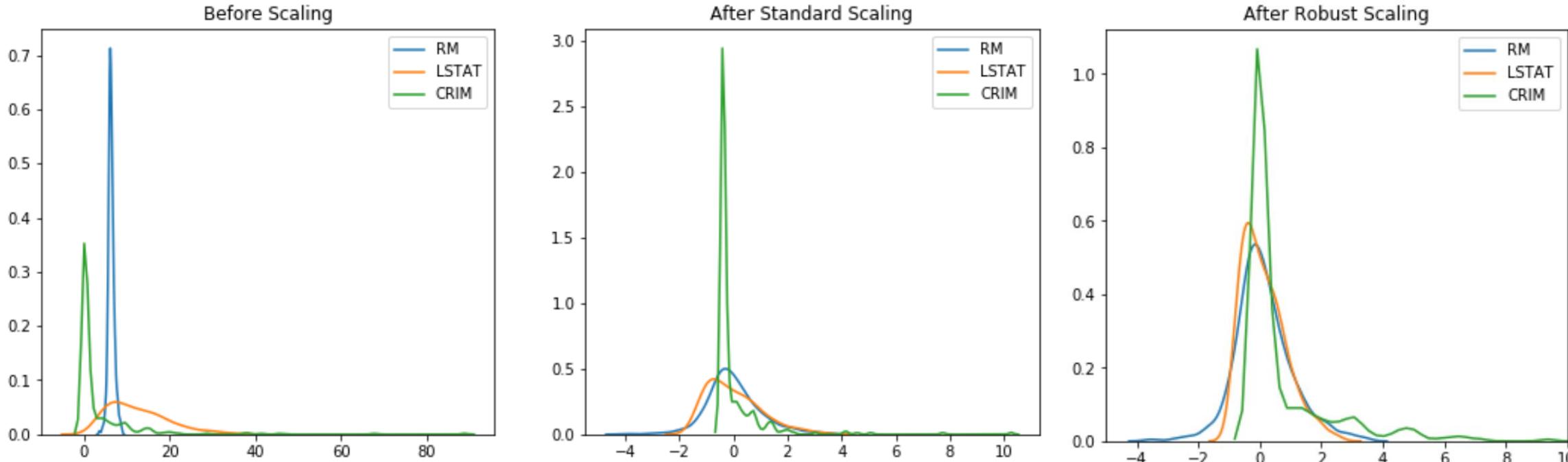


• Robust Scaling: summary

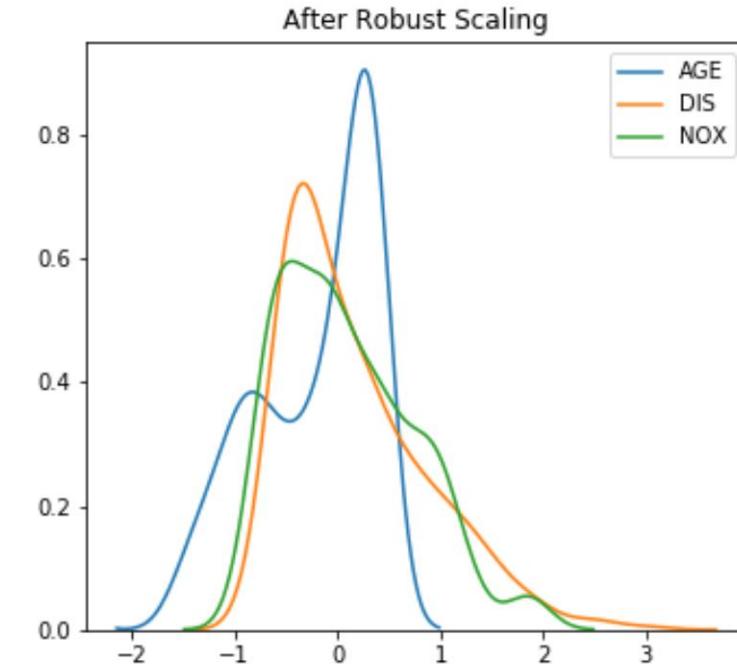
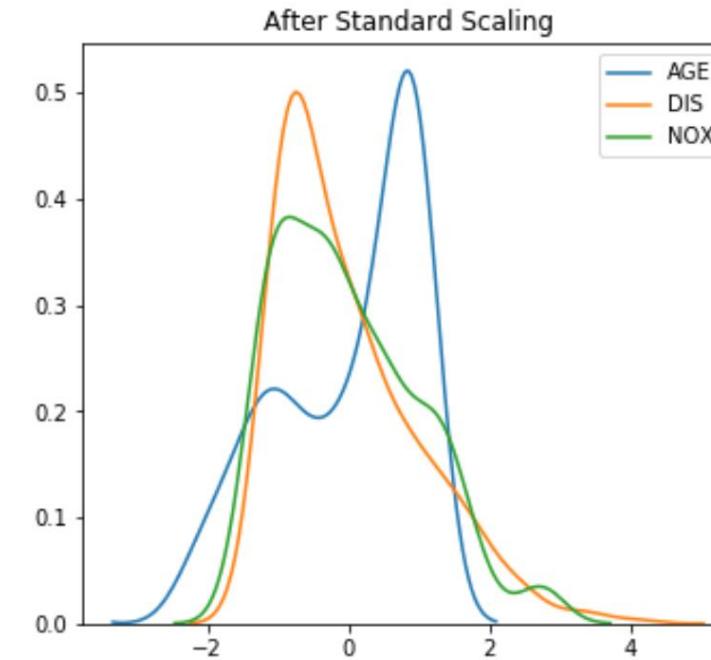
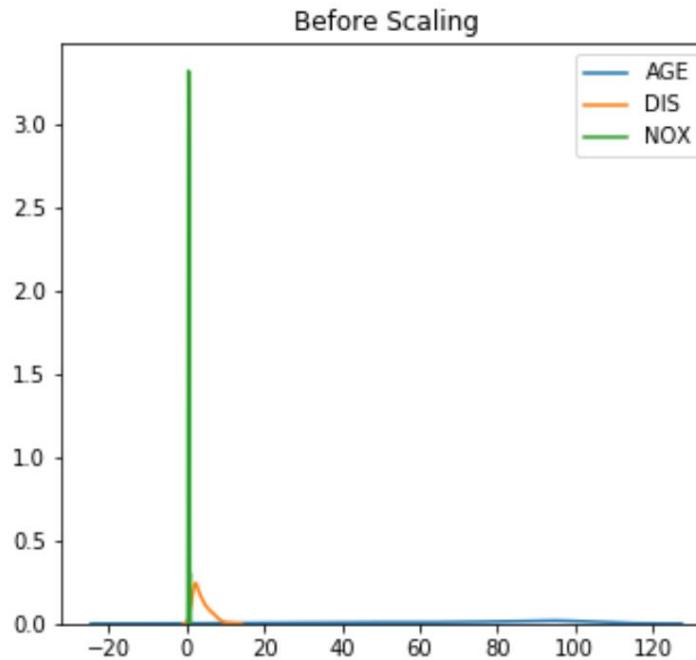
- Median centred at zero
- Handles outliers



• Robust Scaling: Notebook



• Robust Scaling: Notebook



• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
- Robust Scaling with Scikit-learn

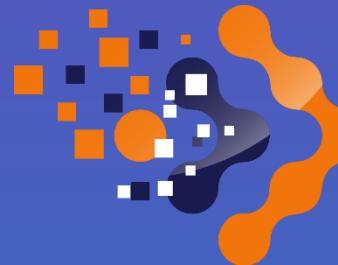




THANK YOU

www.trainindata.com





Train In Data

Scaling to vector unit norm

Scaling to unit norm

In scaling to unit norm, we divide each **feature vector** by either the Manhattan distance (ℓ_1 norm) or the Euclidean distance of the vector (ℓ_2 norm).

$$\ell_1(X) = |x_1| + |x_2| + \dots + |x_n|$$

$$\ell_2(X) = \text{sqr}(x_1^2 + x_2^2 + \dots + x_n^2)$$

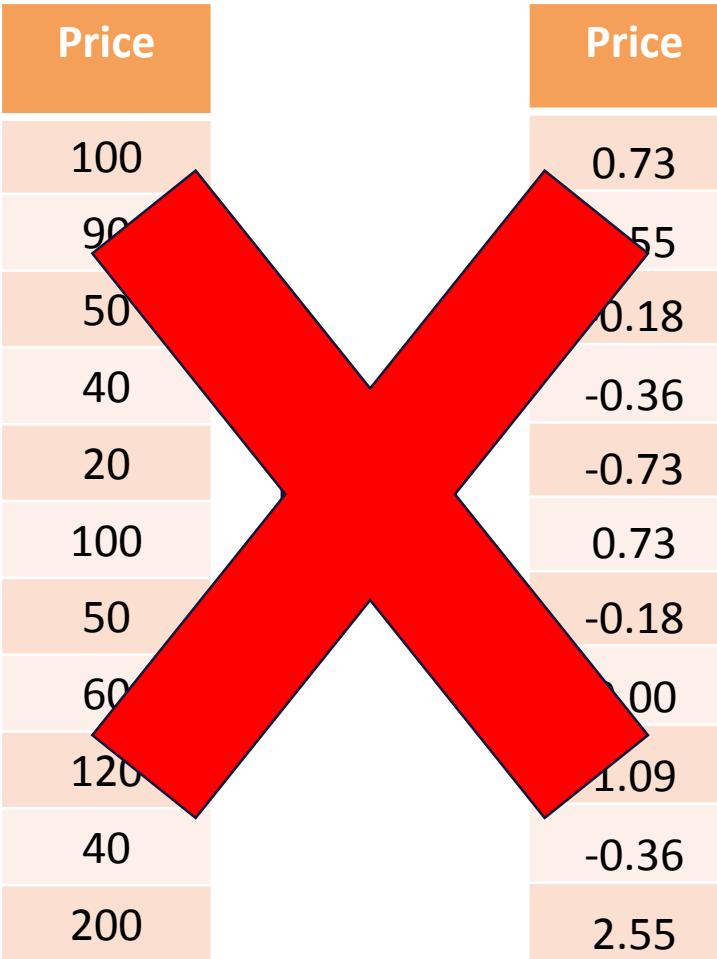


Scaling to unit norm: across features

Price	Price
100	0.73
90	0.55
50	-0.18
40	-0.36
20	-0.73
100	0.73
50	-0.18
60	0.00
120	1.09
40	-0.36
200	2.55



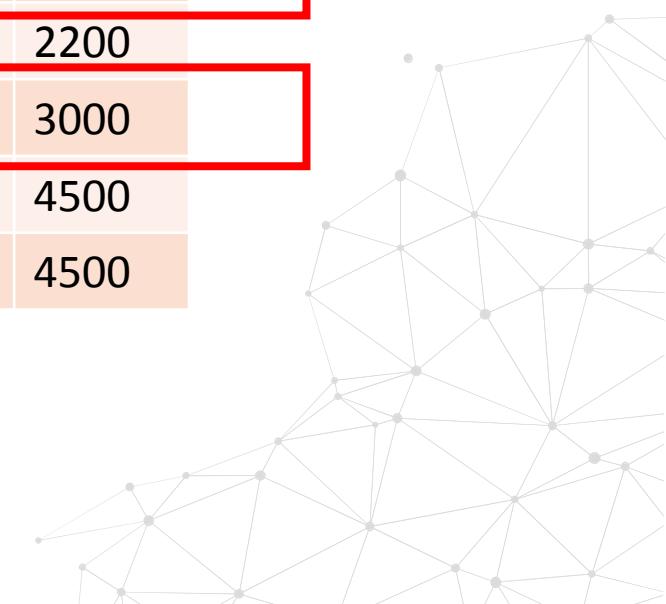
Scaling to unit norm: across features



Scaling to unit norm: across features

Price	Price
100	0.73
90	-0.55
50	0.18
40	-0.36
20	-0.73
100	0.73
50	-0.18
60	0.00
120	1.09
40	-0.36
200	2.55

Gender	Price	Make	Engine
1	100	1	2000
0	90	1	2000
0	50	2	1500
0	60	2	2200
1	3	3	3000
1	120	4	4500
0	200	4	4500



Scaling to unit norm: example

Gender	Price	Make	Engine
1	100	1	2000
0	90	1	2000
0	50	2	1500
0	60	2	2200
1	3	3	3000
1	120	4	4500
0	200	4	4500

Norm of each vector



L2 Norm

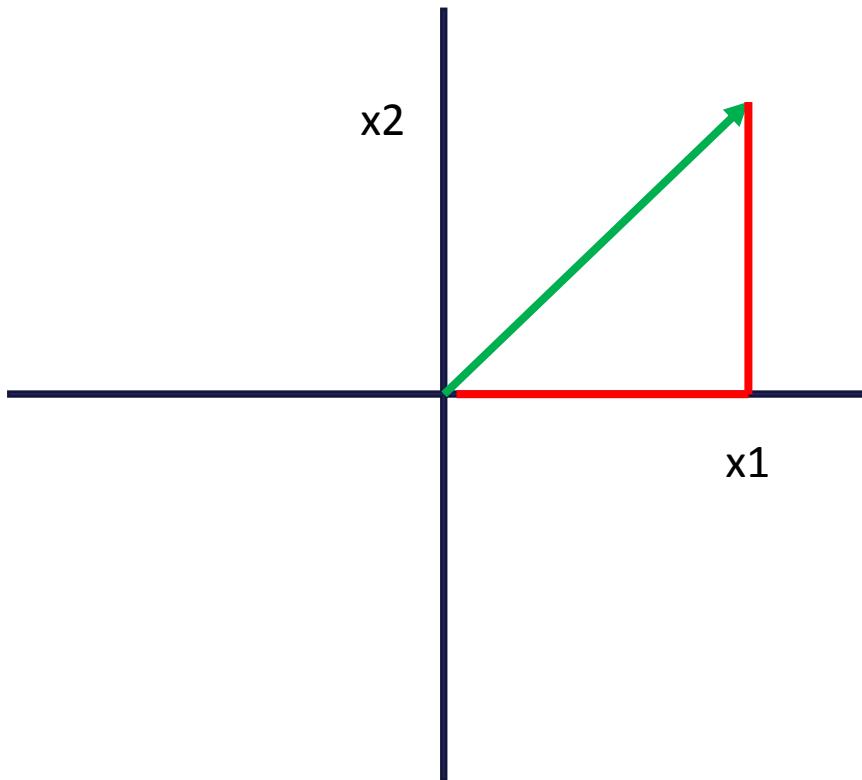
2002
2002
1501
2201
3000
4502
4504

Divide by the norm



Gender	Price	Make	Engine
0.000	0.050	0.000	0.999
0.000	0.045	0.000	0.999
0.000	0.033	0.001	0.999
0.000	0.027	0.001	1.000
0.000	0.001	0.001	1.000
0.000	0.027	0.001	1.000
0.000	0.044	0.001	0.999

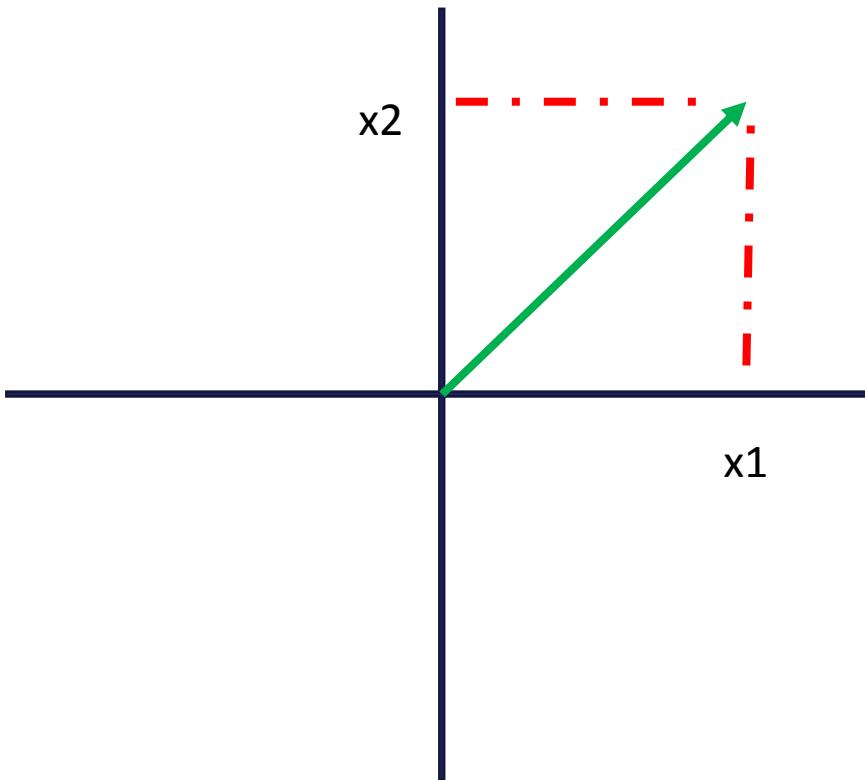
• Manhattan distance, l1



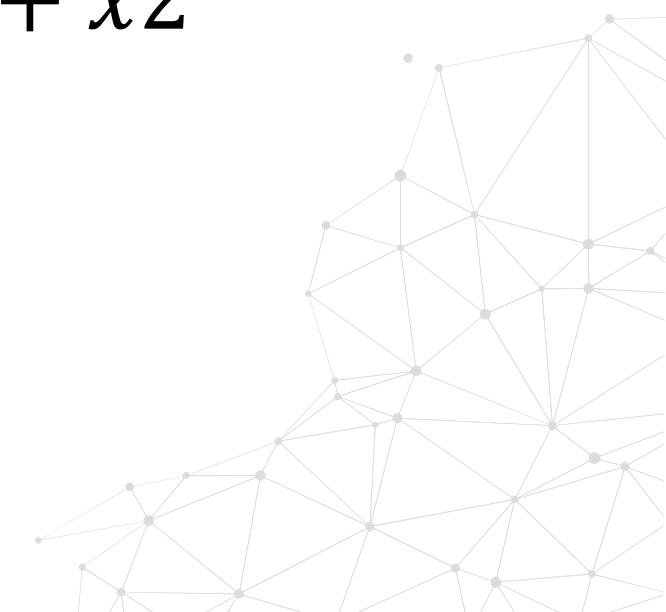
$$L1 = |x_1| + |x_2|$$

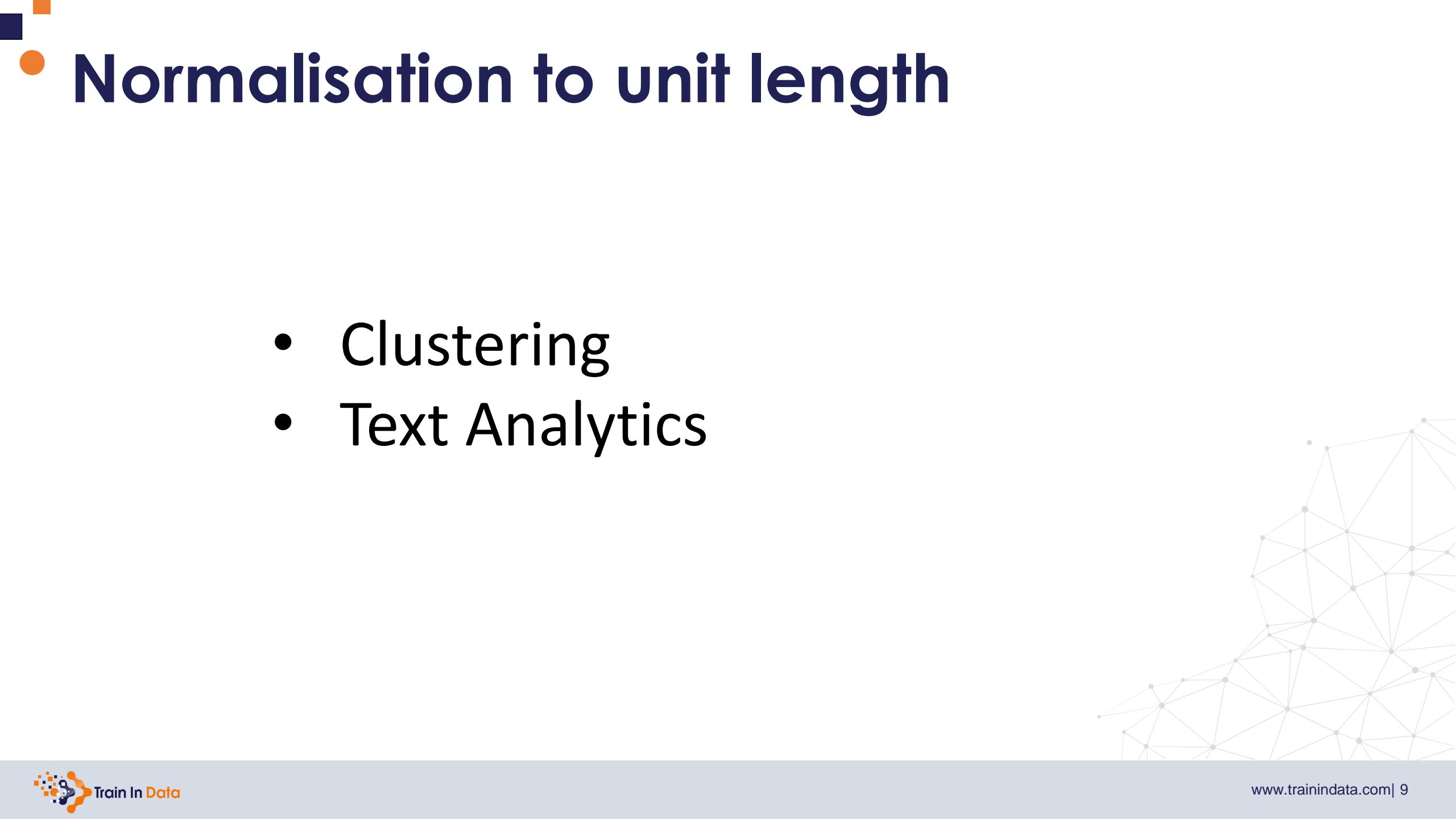


Euclidean distance, l2



$$L2 = \sqrt{x1^2 + x2^2}$$





Normalisation to unit length

- Clustering
- Text Analytics

• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
- Scaling to unit norm with Scikit-learn





THANK YOU

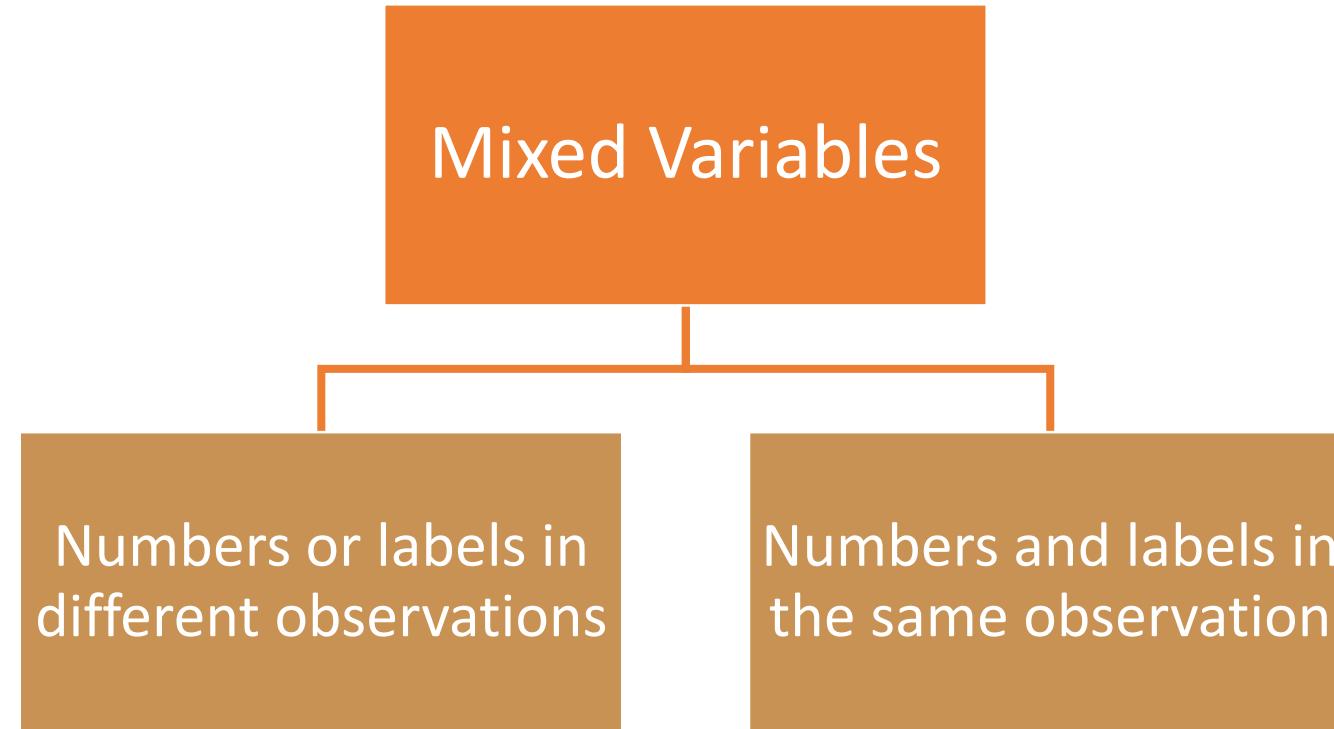
www.trainindata.com





Engineering Variables of Mixed Type

Mixed Variables



Mixed Variables: case 1

Observations show either numbers or categories among their values

- Number of credit accounts (1-100, U, T, M)
 - U = unknown, T = unverified, M = unmatched)
- Number of missed payments (1-3, D, A)
 - D = defaulted, A = arrangements



Mixed variables: case 1

Payments
0
1
0
A
3
D
0



Cat	Num
Nan	0
Nan	1
Nan	0
A	Nan
Nan	3
D	Nan
Nan	0

Mixed Variables: case 2

Observations show both numbers and categories in their values

- Cabin (Titanic) (A15, B18, ...)
- Ticket (Titanic) (A103349)
- Vehicle registration (AB500)
- Postcode (SE18)



Mixed variables: case 2

Cabin
A01
B22
C33
A21
C02
D15
E07



Cat	Num
A	1
B	22
C	33
A	21
C	2
D	15
E	7

• Accompanying Jupyter Notebook



- Read the accompanying Jupyter Notebook
- Download the sample dataset attached to this lecture.





THANK YOU

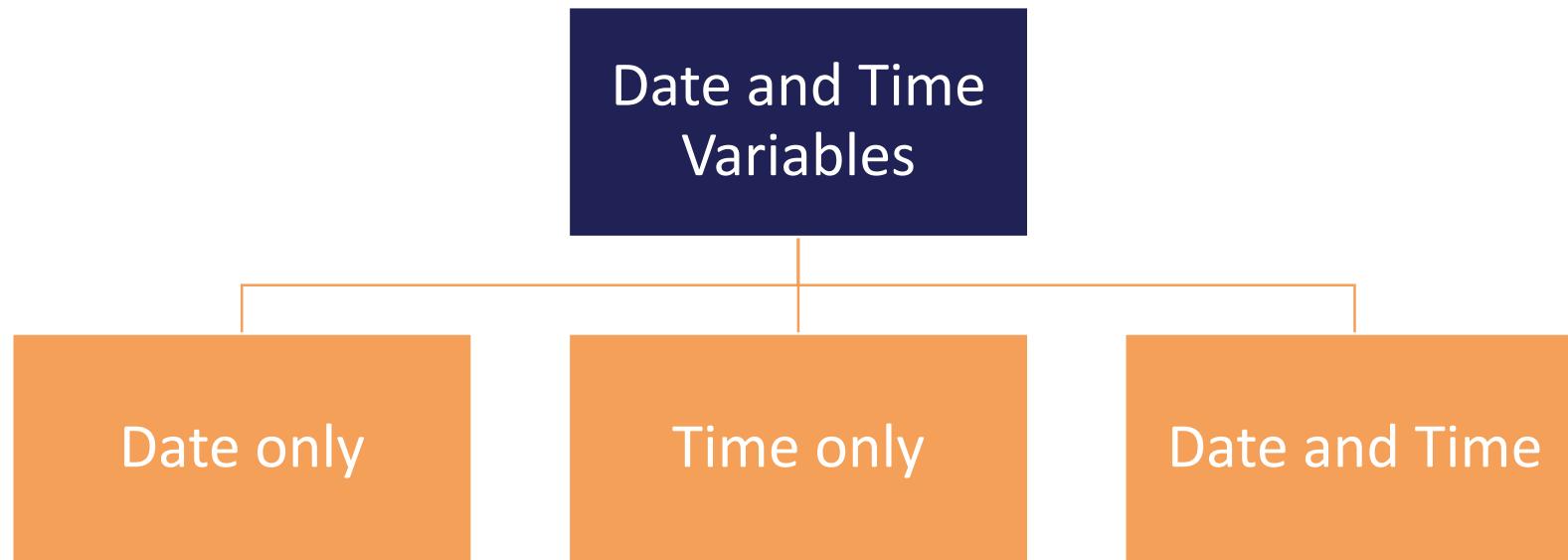
www.trainindata.com





Date and Time Variables

Date and Time Variables



Date and Time Variables

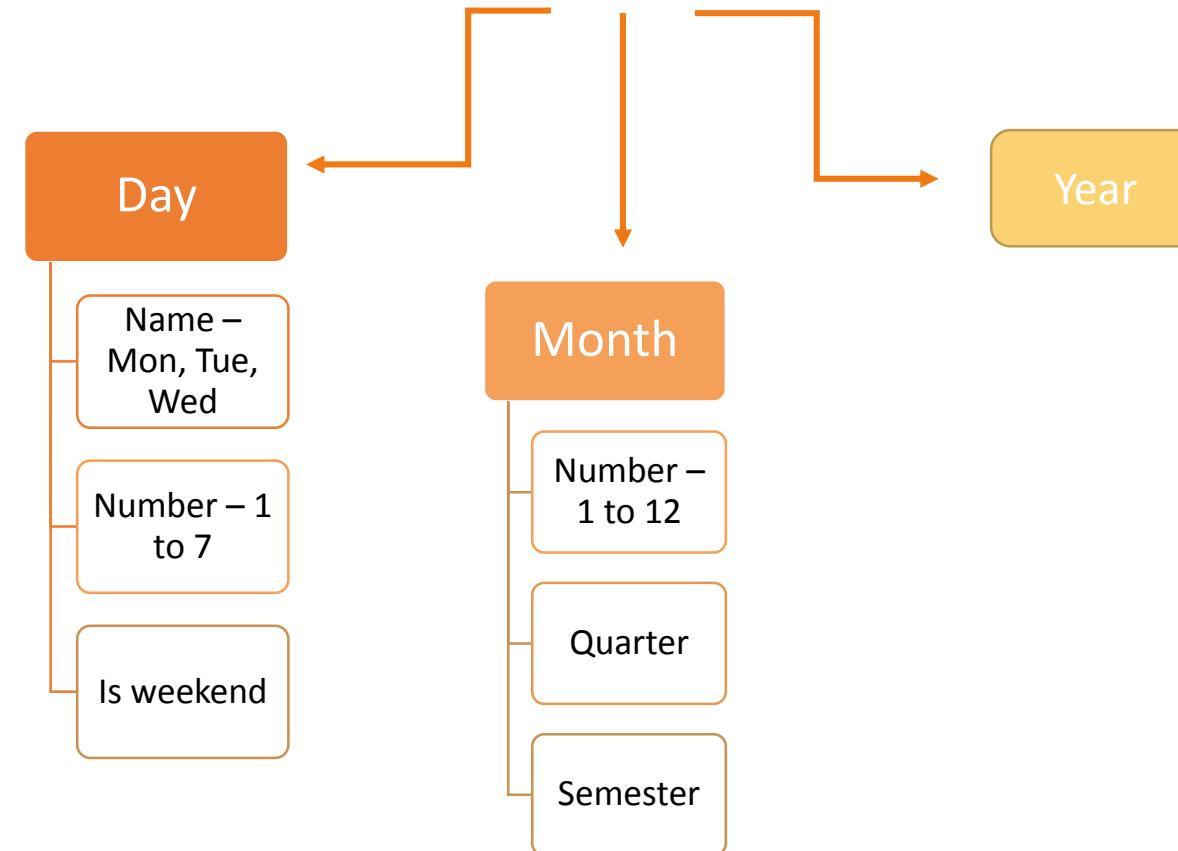
Date and time, or datetime variables they take dates and / or time as values.

- Date of birth ('29-08-1987', '12-01-2012')
- Date of application ('2016-Dec', '2013-March')
- Time of accident (12:20:45)
- Payment date ('29-08-1987 15:20.20')



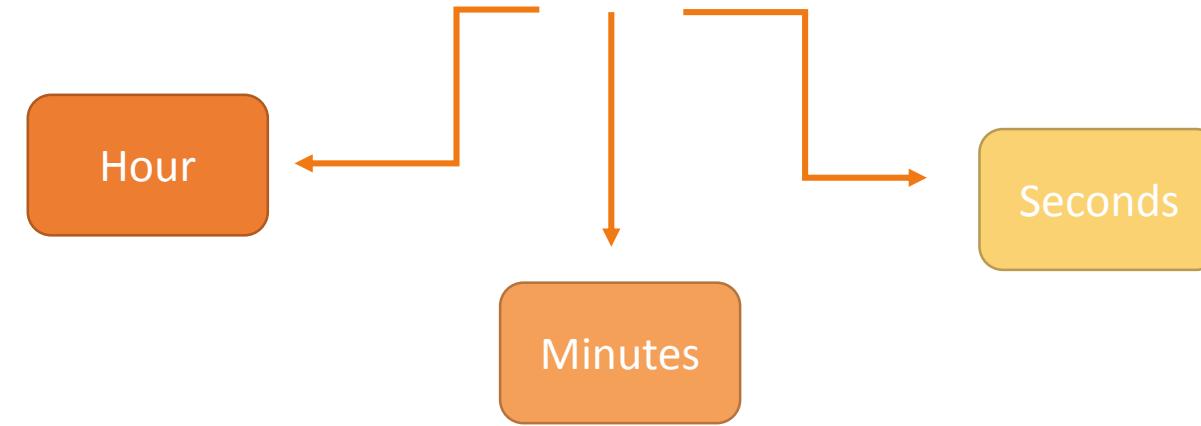
The anatomy of a datetime variable

Payment date ('29-08-1987 15:20.20')



The anatomy of a datetime variable

Payment date ('29-08-1987 15:20.20')



• Elapsed time: time difference

First Payment date ('29-08-1987 15:20.20')

Last Payment date ('29-10-1993 15:20.20')

Date
difference

In days

In months

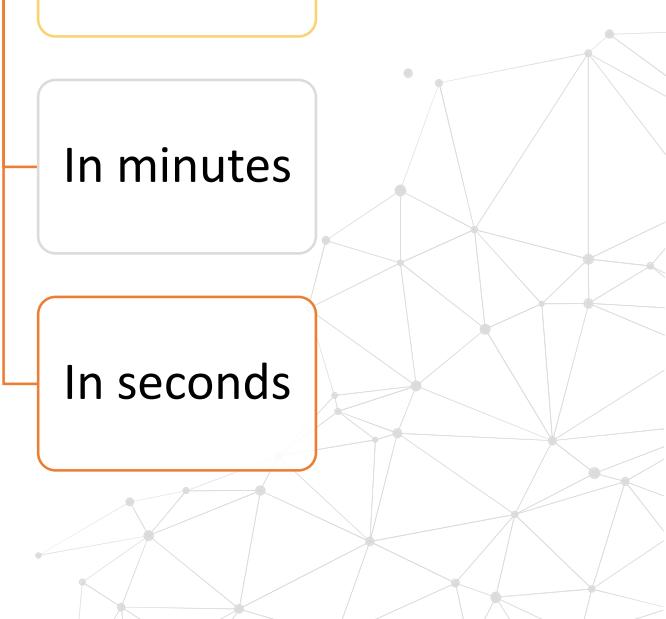
In years

Time
difference

In hrs

In minutes

In seconds



Time zones

Payment date 1 ('29-08-1987 15:20.20⁺⁰²')

Payment date 2 ('29-10-1993 15:20.20⁺⁰⁵')



Payment date 1 ('29-08-1987 13:20.20⁺⁰⁰')

Payment date 2 ('29-10-1993 10:20.20⁺⁰⁰')



• Accompanying Jupyter Notebook



- Engineering dates
- Engineering times





THANK YOU

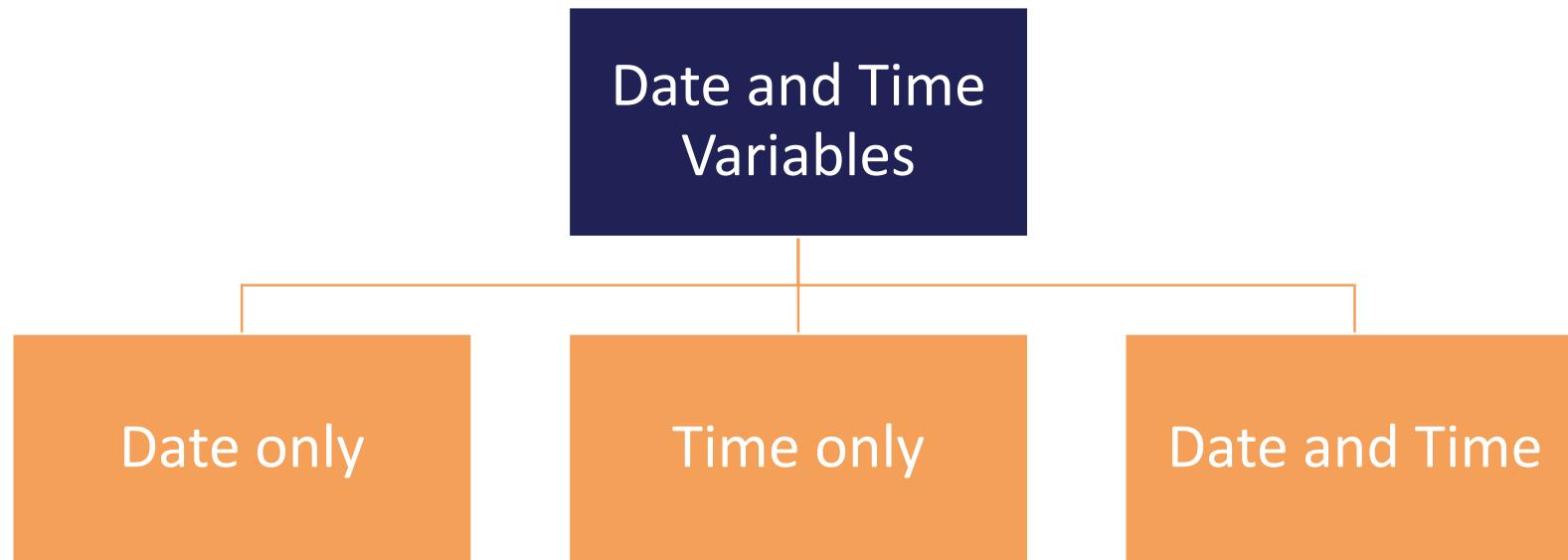
www.trainindata.com





Date and Time Variables

Date and Time Variables



Date and Time Variables

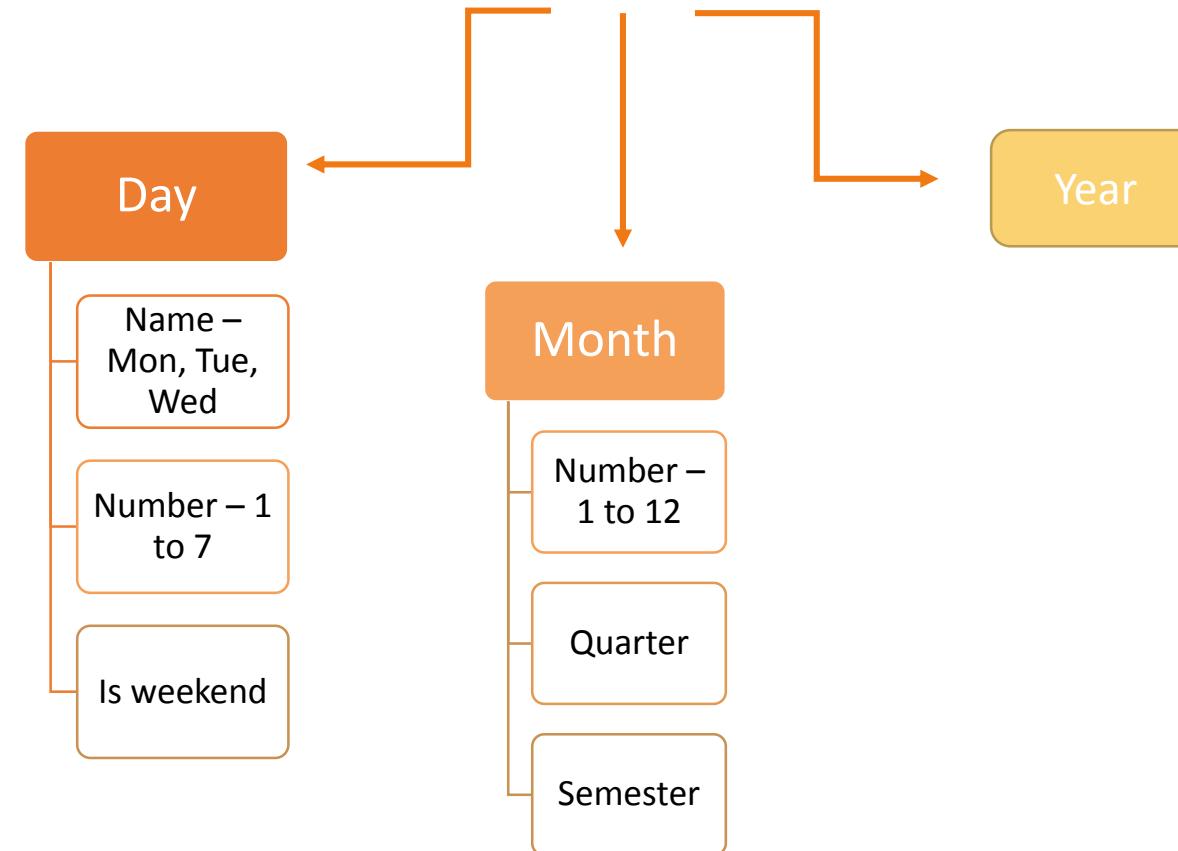
Date and time, or datetime variables they take dates and / or time as values.

- Date of birth ('29-08-1987', '12-01-2012')
- Date of application ('2016-Dec', '2013-March')
- Time of accident (12:20:45)
- Payment date ('29-08-1987 15:20.20')



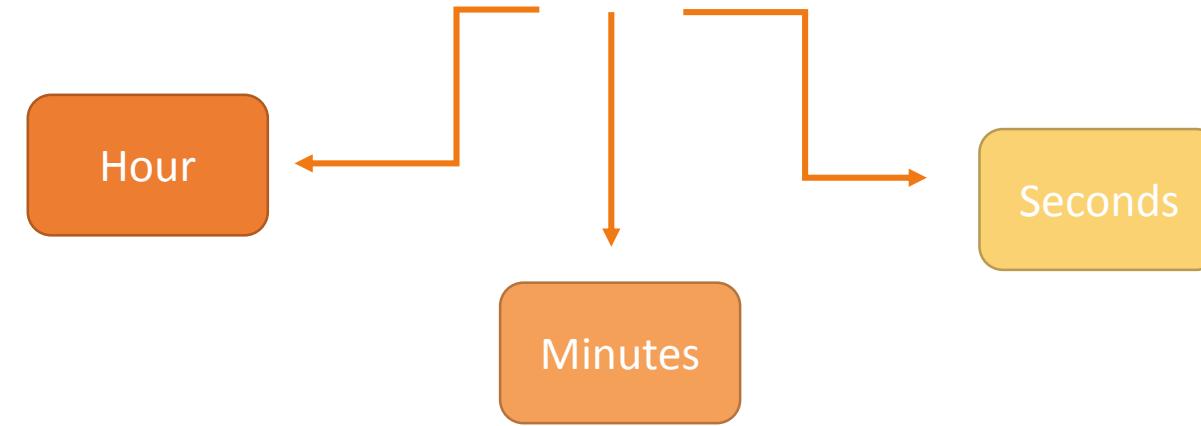
The anatomy of a datetime variable

Payment date ('29-08-1987 15:20.20')



The anatomy of a datetime variable

Payment date ('29-08-1987 15:20.20')



• Elapsed time: time difference

First Payment date ('29-08-1987 15:20.20')

Last Payment date ('29-10-1993 15:20.20')

Date
difference

In days

In months

In years

Time
difference

In hrs

In minutes

In seconds



Time zones

Payment date 1 ('29-08-1987 15:20.20⁺⁰²')

Payment date 2 ('29-10-1993 15:20.20⁺⁰⁵')



Payment date 1 ('29-08-1987 13:20.20⁺⁰⁰')

Payment date 2 ('29-10-1993 10:20.20⁺⁰⁰')



• Accompanying Jupyter Notebook



- Engineering dates
- Engineering times





THANK YOU

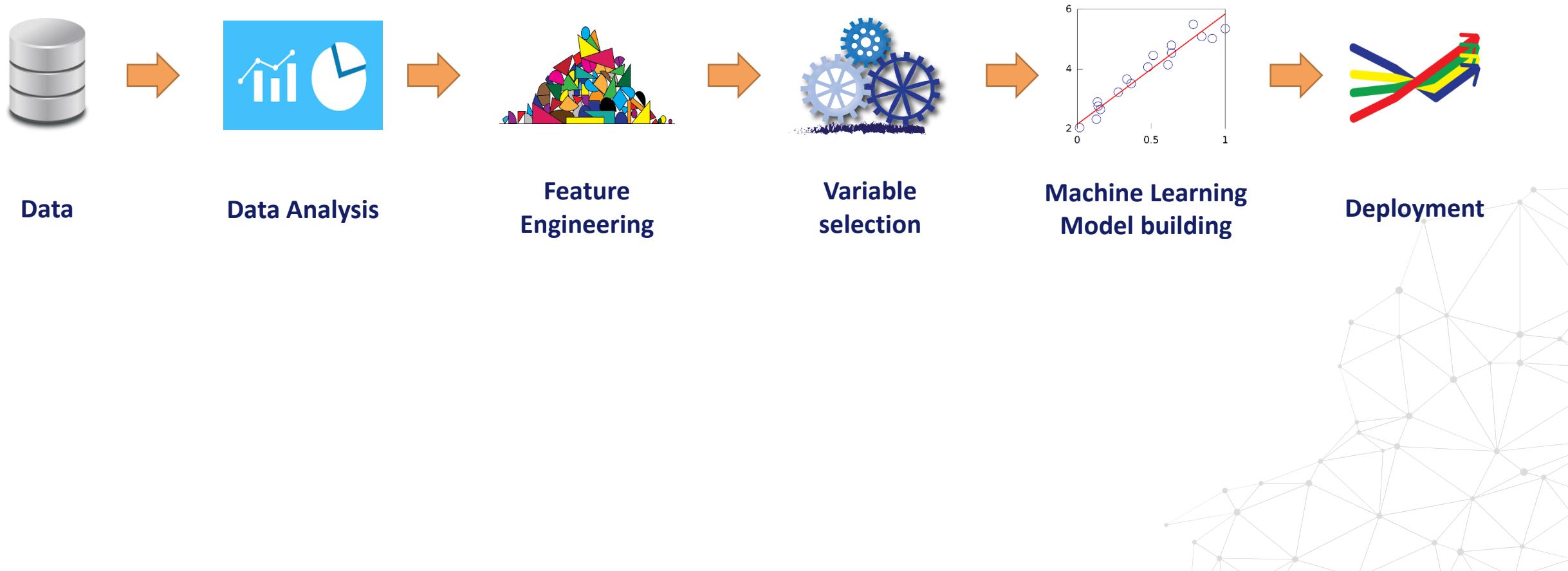
www.trainindata.com





Putting it all together

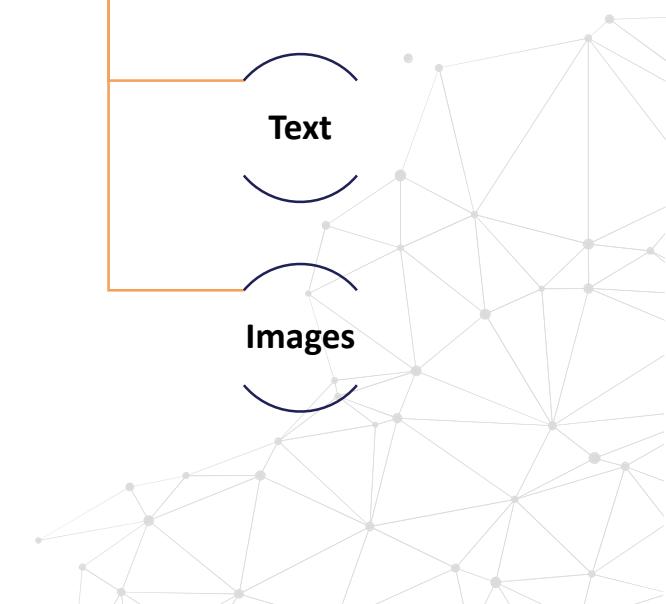
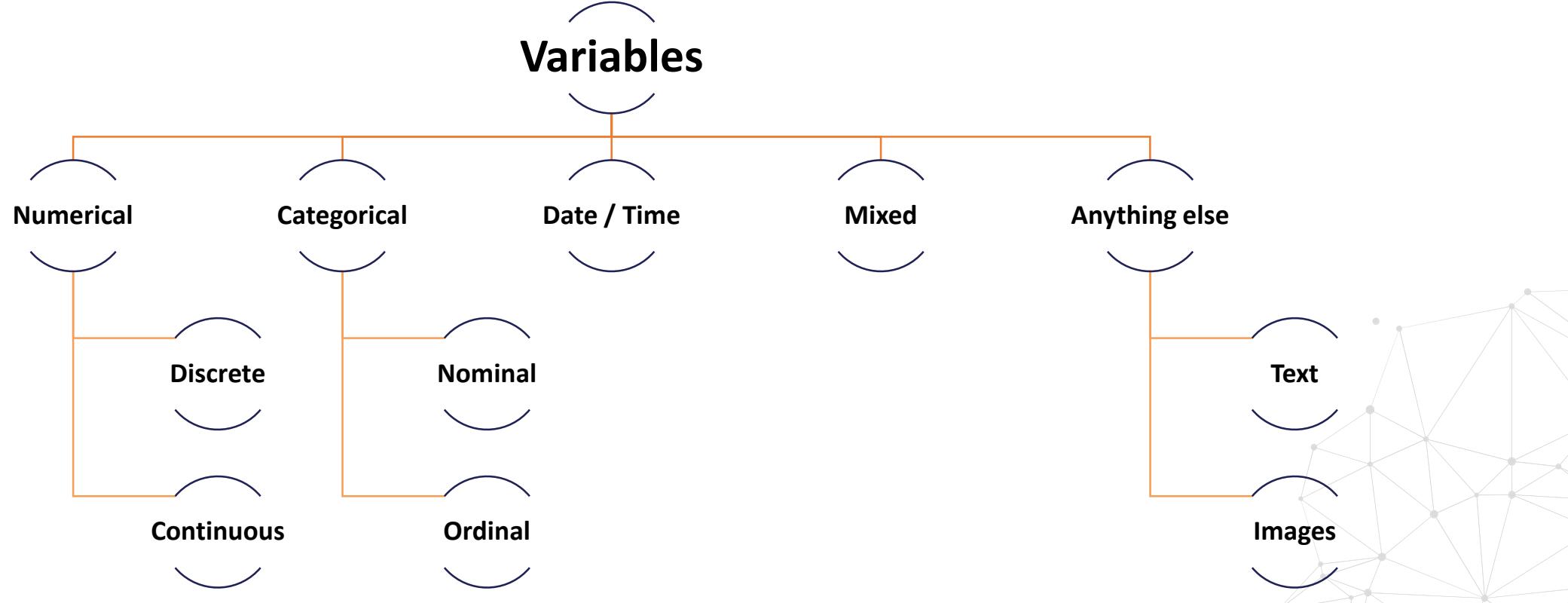
Typical machine learning process



Data Analysis → Variable Types



Data Analysis



Data Analysis → Variable Characteristics



Data Analysis

LINEAR MODEL ASSUMPTIONS

Do the variables fulfill those assumptions?

03

CATEGORICAL VARIABLES

Variables contain strings instead of numbers

02

MISSING DATA

Lack of information for some observations within a variable.

01

Variable
Characteristics

DISTRIBUTIONS

Normal, skewed and others.

04

OUTLIERS

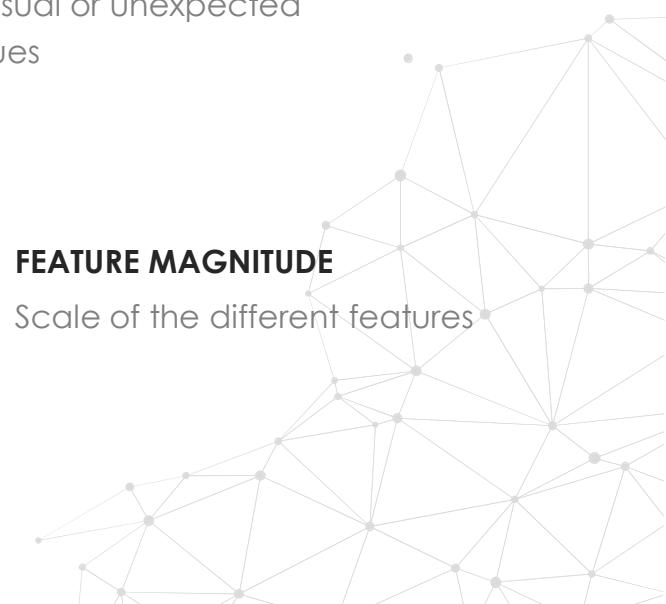
Unusual or unexpected values

05

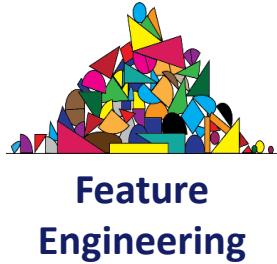
FEATURE MAGNITUDE

Scale of the different features

06



• Feature engineering steps



- **Feature Creation:**
 - Extracting Features from Dates
 - Extracting Features from Mixed Variables
 - Missing Data Imputation
 - Categorical Variable Encoding
 - Numerical Variable Transformation
 - Discretisation
 - Outlier Handling
 - Feature Scaling
 - And more...



Jupyter notebook



- Hard to understand
- Hard to reproduce
- Hard to score new data
- Hard to deploy

The screenshot shows a Jupyter Notebook interface with several code cells and their corresponding outputs:

- Contents [•] & in**: A sidebar menu with sections like Regression, Classification, and House Prices dataset.
- In [1]:** A code cell showing the inspection of numerical variables.

```
numerical = [var for var in numerical if var not in discrete and var not in ['Id', 'SalePrice']] and var not in year_vars]
```

There are 18 numerical and continuous variables

Perfect! Now we have inspected and have a view of the different types of variables that we have in the house price dataset. Let's move on to understand the types of problems that these variables have.
- 3.3 Types of problems within the variables (section 3)**
- 3.3.1 Missing values**
- In [2]:** A code cell showing missing values across all columns.

```
# Let's output variables with NA and the percentage of NA
```

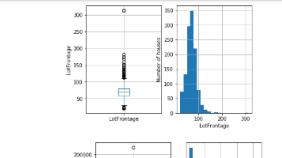
Column	Missing Values (%)
LotFrontage	0.1773976027379726
Alley	0.370711287171212
LotArea	0.0000000000000001
OverallQual	0.0000000000000001
OverallCond	0.0000000000000001
MiscVal	0.0000000000000001
BsmtFinType1	0.0000000000000001
BsmtFinType2	0.0000000000000001
BsmtExposure	0.0000000000000001
BsmtFinSF	0.0000000000000001
BsmtFullBath	0.0000000000000001
BsmtHalfBath	0.0000000000000001
BsmtUnfSF	0.0000000000000001
FireplaceQu	0.026027917260279174
GarageCars	0.0000000000000001
GarageFloor	0.0000000000000001
GarageQual	0.05547845204745452
GarageType	0.0000000000000001
PoolQC	0.99285274258548
Fence	0.89734246573425
MiscFeature	0.303168930337
- 3.3.2 Outliers and distributions**
- In [3]:** A code cell for creating boxplots and histograms to identify outliers in continuous variables.

```
# Let's make boxplots to visualize outliers in the continuous variables
```

Let's make histograms to get an idea of the distribution

```
# for var in numerical:
```

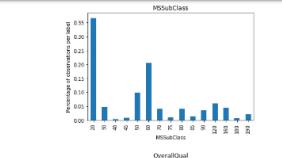
Output: Two plots showing boxplots and histograms for 'LotFrontage' and 'TotalBsmtSF'. Both plots show significant outliers.



The majority of the continuous variables seem to contain outliers. In addition, the majority of the variables are not normally distributed. As we are planning to build linear regression, we need to tackle these to improve the model performance. To tackle the 2 aspects together, I will do discretisation. I will do discretisation with encoding of the intervals following the target mean, as we do in the Discretisation plus encoding lecture in section 8.
- 3.3.3 Outliers in discrete variables**
- In [4]:** A code cell for identifying outliers in discrete variables by calculating the percentage of observations per label.

```
# Now, let's identify outliers in the discrete variables. I will call outliers those values that are present in less than 5 % of the houses. This is exactly the same as finding rare labels in categorical variables. Discrete variables can be pre-processed / engineered as if they were categorical. Keep this in mind
```

Output: A bar chart titled 'MSSubClass' showing the percentage of observations per label. Labels with low percentages are highlighted.



Most of the discrete variables show values that are shared by a tiny proportion of houses in the dataset.
- 3.4 Monotonicity between discrete variables and target values**
- In [5]:** A code cell for plotting the median sale price per value of a discrete variable.

```
# Let's plot the median sale price per value of the discrete
```

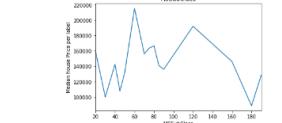
```
# variable
```

```
# for var in discrete:
```

```
    data.groupby(var)[['SalePrice']].median().plot()
```

```
    plt.ylabel('Median House Price per label')
```

Output: A line plot titled 'MSSubClass' showing the median house price per label. The plot shows a clear non-monotonic trend.



• Pipeline



- Clear
- Concise
- Reproducible
- Able to score new data
- Easy to deploy



Pipeline

```
price_pipe = pipe([
    # add a binary variable to indicate missing information for the 2 variables below
    ('continuous_var_imputer', msi.AddNaNBinaryImputer(variables = ['LotFrontage', 'GarageYrBlt'])),  

    # replace NA by the median in the 3 variables below, they are numerical
    ('continuous_var_median_imputer', msi.MeanMedianImputer(imputation_method='median', variables = ['LotFrontage', 'GarageYrBlt', 'MasVnrArea'])),  

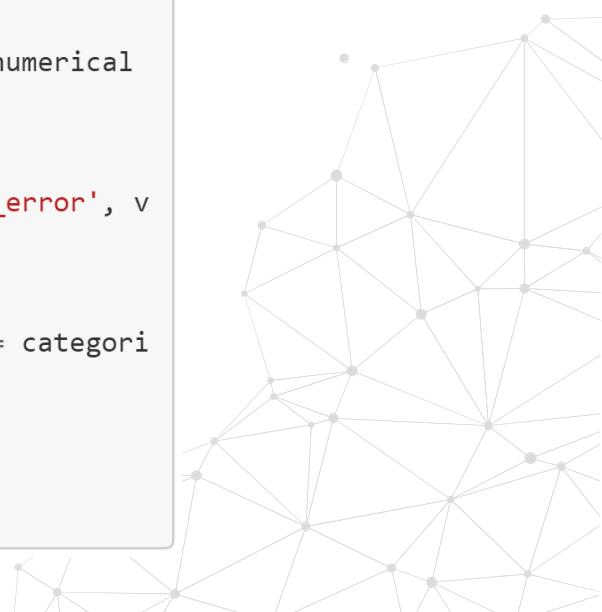
    # replace NA by adding the label "Missing" in categorical variables (transformer will skip those variables where there is no NA)
    ('categorical_imputer', msi.CategoricalVariableImputer(variables = categorical)),  

    # there were a few variables in the submission dataset that showed NA, but these variables did not show NA in the train set.
    # to handle those, I will add an additional step here
    ('additional_median_imputer', msi.MeanMedianImputer(imputation_method='median', variables = numerical)),  

    # discretise numerical variables using trees
    ('numerical_tree_discretiser', dsc.DecisionTreeDiscretiser(cv = 3, scoring='neg_mean_squared_error', variables = numerical, regression=True)),  

    # remove rare labels in categorical and discrete variables
    ('rare_label_encoder', ce.RareLabelCategoricalEncoder(tol = 0.03, n_categories=1, variables = categorical+discrete)),  

    # encode categorical variables using the target mean
    ('categorical_encoder', ce.MeanCategoricalEncoder(variables = categorical+discrete))
])
```



• Pipeline

```
# train the pipeline  
price_pipe.fit(X_train, y_train)  
  
# score data  
price_pipe.transform(X_train)  
price_pipe.transform(X_test)  
  
price_pipe.transform(live_data)
```





THANK YOU

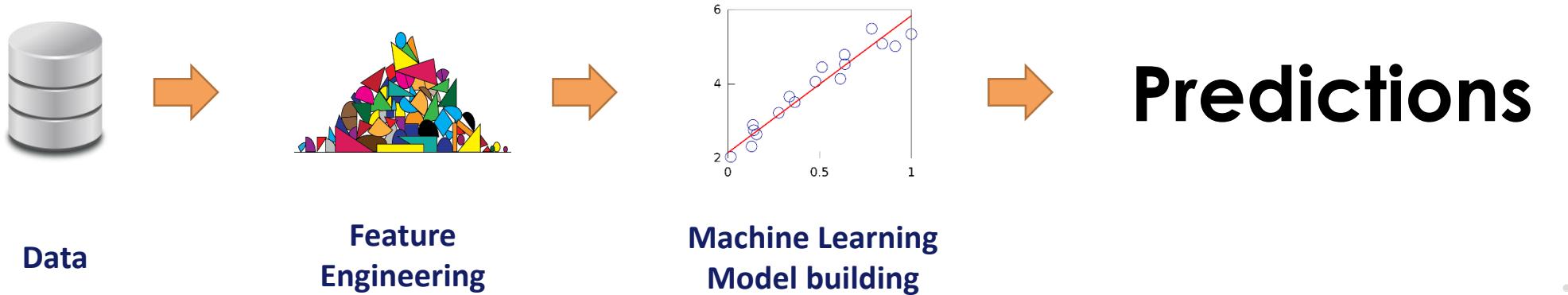
www.trainindata.com





Assembling a Pipeline

Assembling a pipeline: steps



• Pipeline steps: training

```
# building the pipeline
imputer = SimpleImputer(strategy = 'mean')
category_encoder = OneHotEncoder()
discretiser = KBinsDiscretizer(strategy='quantile')
scaler = StandardScaler()

train_transformed_1 = imputer.fit_transform(X_train)
train_transformed_2 = category_encoder.fit_transform(train_transformed_1)
train_transformed_3 = discretiser.fit_transform(train_transformed_2)
train_final = scaler.fit_transform(train_transformed_3)

model = GradientBoostingClassifier()

model.fit(train_final)

train_pred = model.predict(train_final)
```



• Pipeline steps: testing

```
# to score the test set
test_transformed_1 = imputer.transform(X_test)
test_transformed_2 = category_encoder.transform(test_transformed_1)
test_transformed_3 = discretiser.transform(test_transformed_2)
test_final = scaler.transform(test_transformed_3)

test_pred = model.predict(test_final)
```



Assembling a Pipeline

Pipeline - class that allows to run transformers and a machine learning model in sequence.

- Most steps are Transformers
- Last step can be an Estimator

```
from sklearn.pipeline import Pipeline

pipeline = Pipeline([
    ('imputation', SimpleImputer(strategy = 'mean')),
    ('encoding', OneHotEncoder()),
    ('discretisation', KBinsDiscretizer(strategy='quantile')),
    ('scaling', StandardScaler()),
    ('model', GradientBoostingClassifier())
])

# to train the model
pipeline.train(X_train, y_train)

# to score the test set
pipeline.predict(X_test)

# to score new data
pipeline.predict(new_data)
```

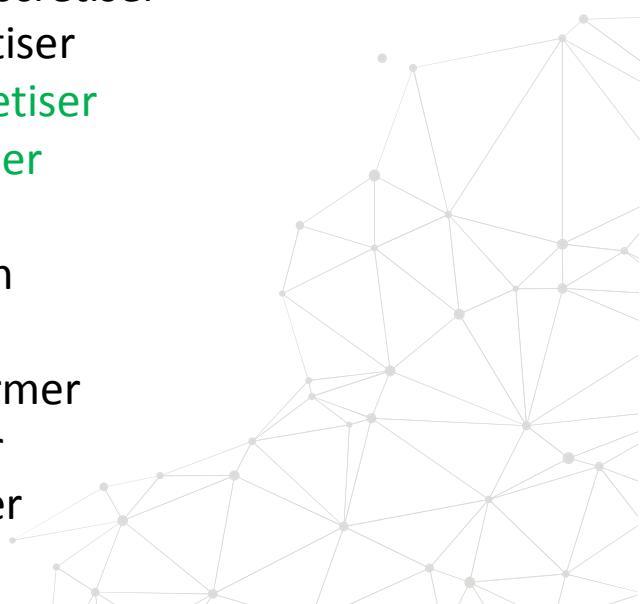
Scikit-learn transformers

- Missing Data Imputation
 - SimpleImputer
- Categorical Variable Encoding
 - OneHotEncoder
 - LabelEncoder
- Discretisation
 - KBinsDiscretizer
- Variable Transformation
 - PowerTransformer
 - FunctionTransformer
- Scaling
 - StandardScaler
 - MinMaxScaler
 - RobustScaler
 - Normalizer



• Feature-engine transformers

- Missing Data Imputation
 - MeanMedianImputer
 - RandomSampleImputer
 - EndTailImputer
 - AddMissingIndicator
 - CategoricalVariableImputer
- Categorical Variable Encoding
 - CountFrequencyCategoricalEncoder
 - OrdinalCategoricalEncoder
 - MeanCategoricalEncoder
 - WoERatioCategoricalEncoder
 - OneHotCategoricalEncoder
 - RareLabelCategoricalEncoder
- Outlier Removal
 - Windsorizer
 - ArbitraryOutlierCapper
 - OutlierTrimmer
- Discretisation
 - EqualFrequencyDiscretiser
 - EqualWidthDiscretiser
 - DecisionTreeDiscretiser
 - UserInputDiscretiser
- Variable Transformation
 - LogTransformer
 - ReciprocalTransformer
 - PowerTransformer
 - BoxCoxTransformer





THANK YOU

www.trainindata.com

