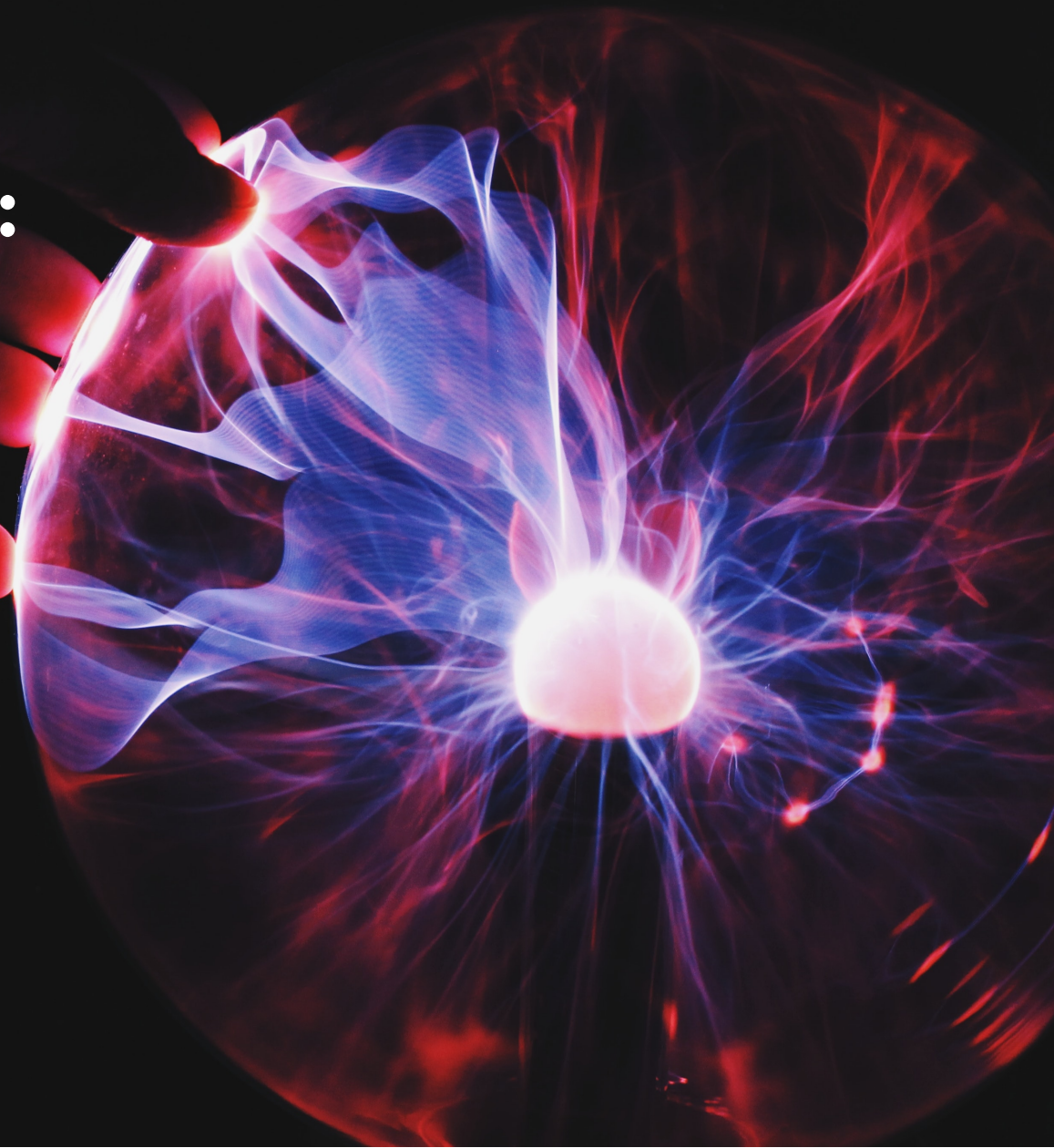
Machine Learning in R: Workshop Series

Feature Engineering

Simon Schölzel

Research Team Berens

2020-08-20 (updated: 2020-09-13)



Agenda

1. Introduction to feature engineering

2. Engineering of categorical (`fct`) and numerical features (`dbl`, `int`)

1. Engineering of categorical features
   - Encoding of ordered and unordered features
   - Aggregation of near-zero variance features
2. Engineering of numerical features
   - 1 : 1 transformations
   - 1 : N transformations
   - N : N transformations
3. Outlier correction
4. Missing values

3. Data leakage

4. Feature selection

1. Intrinsic Methods
2. Filter and Wrapper Methods

5. Outlook

Learning Objectives 💡

In this workshop you will learn about the basic techniques of feature engineering. You will familiarize yourself with a toolbox of several numerical as well as visual techniques for inspecting your feature set and eventually improving the predictive performance of your machine learning models.

More specifically, after this workshop you will

- be familiar with the broad field of feature engineering, including feature transformation, feature extraction and feature selection,
- be able to detect potential pitfalls in the representation of your categorical (`fct`) and numerical (`int`, `dbl`) features and know how to alleviate them,
- know important techniques for investigating and handling outliers as well as missing values,
- be aware of the concept of data leakage and how to prevent it, and
- have a good overview of the different approaches to feature selection.

Introduction to Feature Engineering

Most machine learning models require the data to be of a certain shape. That is, even small perturbation in your features can adversely affect the performance of your model. For example:

Differently scaled features

- linear regression (scale absorbed by coefficients)
- distance-based learners (e.g., SVM or kNN)

Highly skewed features

- non-parametric learners (e.g., tree-based methods)
- parametric learners which assume normality (e.g., linear regression)

Outlier, i.e. extreme data points

- trees-based methods
- linear regression

Features that contain missing values

- e.g. CART decision trees (surrogate splits)
- e.g. C4.5 decision trees or certain RF implementations

Multicollinearity among features

- e.g. PLS, PCA, penalized regression
- e.g. linear regressions or neural networks

Irrelevant / uninformative features

- stepwise selection and penalized regression
- k-NN (computational cost grow exponentially)

Introduction to Feature Engineering

> [F]eature engineering - the process of creating representations of data that increase the effectiveness of a model. ~ Kuhn, M./Johnson, K. (2019)

The process of ***feature engineering*** subsumes several interrelated steps, such as

- applying transformations to the data (**feature transformation**),
- creating new features based on preexisting features (**feature extraction**), and
- omitting redundant features and retaining informative features (**feature selection**).

Moreover, the process of feature engineering may be performed in a *supervised* (i.e. data-driven) or *unsupervised* (e.g., theory- or knowledge-based) manner.

Introduction to Feature Engineering

Introductory Example

Have you ever thought about the numerous potential ways of how to represent a date feature (`2020-09-09`) in your machine learning pipeline? There are more options than you would think at first glance...

Option 1: Number of days since *unix time* (`1970-01-01`)

```
> Time difference of 18514 days
```

Option 2: Year, month and day of the week feature

```
> [1] 2020    9    4
```

Option 3: Day of the year

```
> [1] 253
```

Option 4: Boolean period indicator (`2020-06-29` < x < `2020-08-11`)

```
> [1] FALSE
```

Engineering of Categorical Features (`fct`)

Encoding of Unordered Features

If a model requires only numerical data as input, categorical features must be transformed into a numerical representation. In the case of an unordered feature with `C` levels, the feature is commonly transformed into a set of `C-1` binary features via **one-hot encoding**.

```
> # A tibble: 3 x 3
>   species   species_gentoo species_chinstrap
>   <fct>              <dbl>             <dbl>
> 1 Adelie                 0                 0
> 2 Gentoo                 1                 0
> 3 Chinstrap              0                 1
```

Note: If your `lm` includes an intercept and you generate `C` dummies, one for each factor level, your final model is overdetermined, i.e. the intercept can always be computed as a linear combination of the `C` dummies. In this case the OLS algorithm may fail as your feature matrix is not invertible.

Engineering of Categorical Features (`fct`)

Encoding of Ordered Features

In the case of ordered factor levels (e.g., `low`, `medium` and `high`), we may resort to the following approaches:

1. One-hot encoding to generate binary dummies - omits information included in relative ordering
2. Representation as a feature of type integer (`int`) with equal intervals between factor levels
3. Encoding via polynomial contrasts to produce a numeric (`dbl`) representation

Linear (**quadratic**) **polynomial contrast** to represent a linear (quadratic) ordering:

```
> # A tibble: 3 x 3
>   level  coef_linear coef_quadratic
>   <fct>        <dbl>          <dbl>
> 1 low          -0.71           0.41
> 2 medium        0             -0.82
> 3 high          0.71           0.41
```

Theoretically, a polynomial contrast can model most non-linear shape in the relationship between the response and the categorical feature of up to degree `C-1`. Yet, in practice the exploration of polynomial contrasts does seldom exceed polynomials of degree 2 [1].

Engineering of Categorical Features (`fct`)

Aggregation of Near-zero Variance Features

If some feature levels occur very rarely, the feature is frequently considered a **near-zero variance feature**. This is oftentimes the case for categorical features of high cardinality (i.e. where `C` is large).

For these imbalanced features, one-hot encoding becomes computationally demanding. Simultaneously, certain feature levels may never realize during resmapling (e.g., cross-validation) which results in a binary predictor containing only zeros in a given split.

Rule-of-thumb for near-zero variance features: `(#zeros / #ones) > 19` [1]

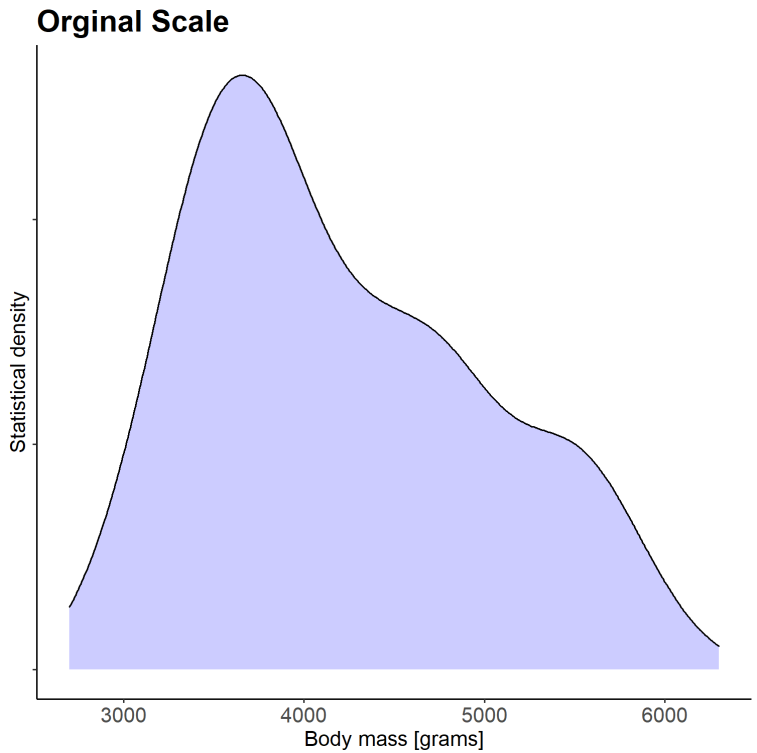
```
> # A tibble: 3 x 3
>   island        n ratio
>   <fct>     <int> <dbl>
> 1 Biscoe      168  1.05
> 2 Dream       124  1.77
> 3 Torgersen    52  5.62
```

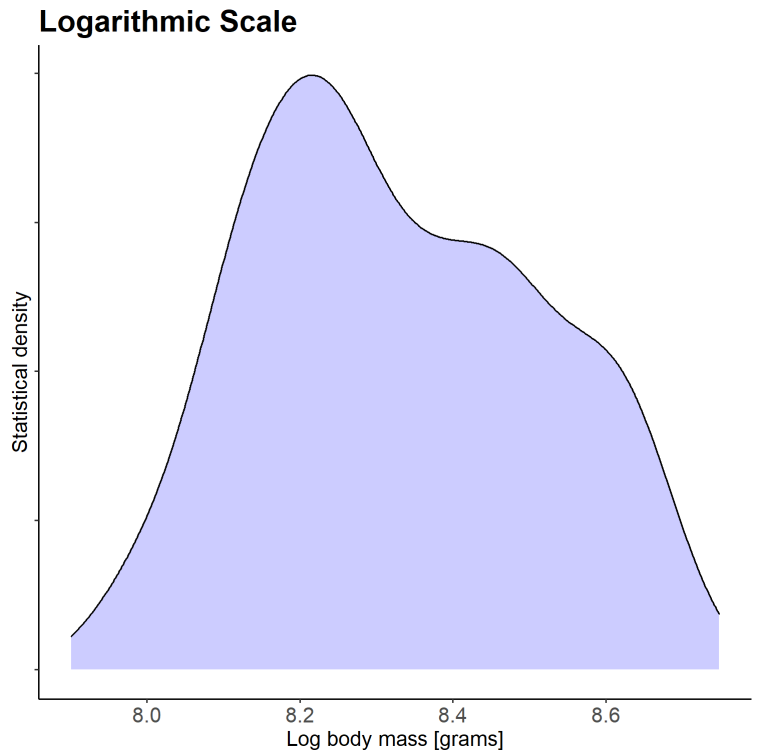
If a feature is considered a near-zero variance feature, a potential remedy is to lump together rare feature levels together into an "other" category (e.g., using `` `forcats::fct_lump()` ``).

Engineering of Numerical Features (`dbl`, `int`)

1 : 1 Transformations

Log-transformation and **Inversion**: Resolves skewness and shifts the distribution closer to a Gaussian.





Note: Some of these transformations are problematic for zero or negative values. In such cases simple corrections are available to address these issues.

Engineering of Numerical Features (`dbl`, `int`)

1 : 1 Transformations

Box-Cox power transformation [1]: Resolves skewness and shifts the distribution closer to a Gaussian.

$$Y^{(\lambda)} = \frac{Y^{\lambda} - 1}{\lambda} \quad for \quad \lambda \neq 0$$

$$Y^{(\lambda)} = ln(Y) \quad for \quad \lambda = 0$$

`λ` is considered the so called *power parameter* which can be estimated via maximum likelihood estimatio or hyperparameter tuning. Certain values for the parameter `λ` correspond to commonly known transformations:

- `λ` = 1: no transformation
- `λ` = 0: log transformation
- `λ` = 0.5: square root transformation
- `λ` = -1: inversion

[1]: **Box, G. E. P./Cox, D. R. (1964):** An Analysis of Transformations. Journal of the Royal Statistical Society, Series B (Methodological), Vol. 26, No. 2 (1964), pp. 211-252.

Engineering of Numerical Features (`dbl`, `int`)

1 : 1 Transformations

z-Normalization: Modifies the scale of the predictor.

$$Z = \frac{X - \mu}{\sigma}$$

The z-Normalization involves two steps: *centering* (substracting the mean) and *scaling* (dividing by the standard deviation). Afterwards the `Z` follows a standard normal distribution with mean zero and standard deviation of 1.

min/max-Normalization: Modifies the scale of the predictor.

$$X' = \frac{X - min(X)}{max(X) - min(X))}$$

The min/max-Normalization transforms the data points to lie on the [0; 1]-interval. A generalization of min/max-Normalization is **range scaling** where the data points are projected onto an arbitrary interval.

Engineering of Numerical Features (`dbl`, `int`)

1 : N Transformations

Basis expansion: Establishes non-linear relationships between a predictor and the response by adding polynoms (e.g., quadratic and cubic terms).

```
penguins %>%
  mutate(
    bill_length_mm = flipper_length_mm^2, #add quadratic term
    bill_length_mm = flipper_length_mm^3) #add cubic term
```

This enables us to specify additive models that account for a potential non-linear relationship in the form of:

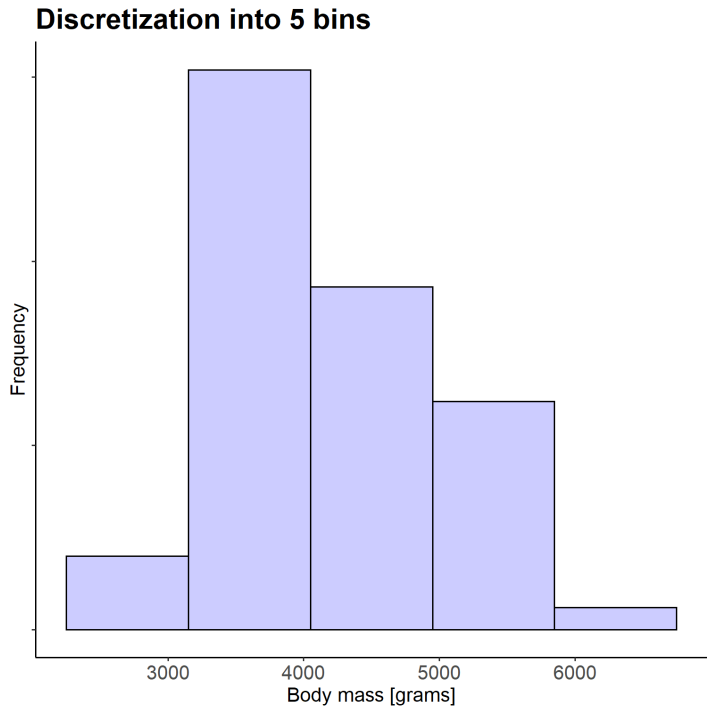
$$Y = X + X^2 + X^3 + \dots$$

*Note: A technique related to basis expansion is the inclusion of **interaction terms** which model non-linear relationship between multiple predictors and the response.*

Engineering of Numerical Features (`dbl`, `int`)

1 : N Transformations

Discretization: Reduces a numeric feature to a set of categorical features.



In certain cases, binning helps to reduce complexity of the relationship between `X` and `Y` and hence to facilitate the interpretation.

However, it should only be used with great care as

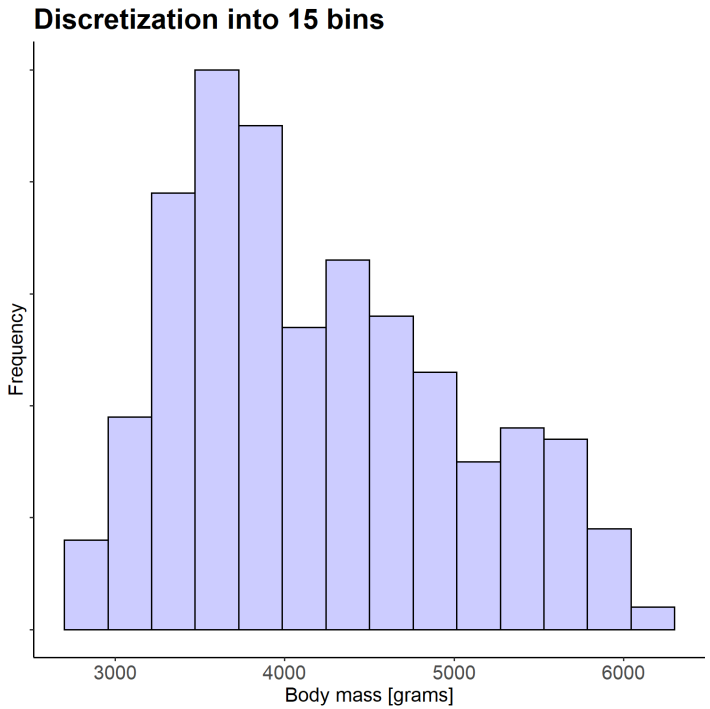
- it is accompanied by an information loss,
- it may obscure the relation between `X` and `Y`,
- bins are often assigned subjectively, and
- the model performance oftentimes suffers.

Note: The `dplyr::cut_width()` function is a good place to start when discretizing numerical predictors.

Engineering of Numerical Features (`dbl`, `int`)

1 : N Transformations

Discretization: Reduces a numeric feature to a set of categorical features.



In certain cases, binning helps to reduce complexity of the relationship between `X` and `Y` and hence to facilitate the interpretation.

However, it should only be used with great care as

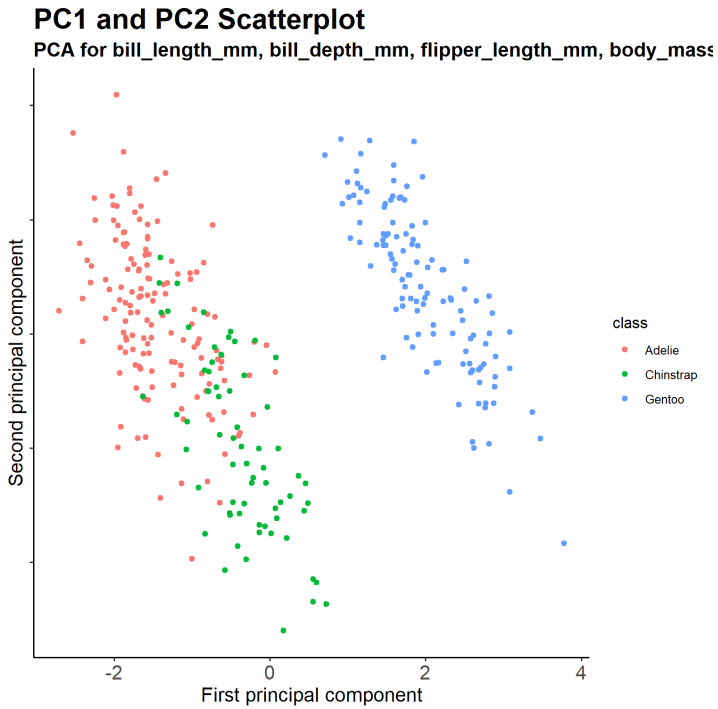
- it is accompanied by an information loss,
- it may obscure the relation between `X` and `Y`,
- bins are often assigned subjectively, and
- the model performance oftentimes suffers.

Note: The `dplyr::cut_width()` function is a good place to start when discretizing numerical predictors.

Engineering of Numerical Features (`dbl`, `int`)

N : N Transformations

Dimensionality reduction techniques: Generate uncorrelated predictors to resolve multicollinearity.



The underlying idea of dimensionality reduction techniques for feature engineering is to extract more informative features from the existing feature set. Usually, this is at the expense of feature interpretability.

Principal Component Analysis (**PCA**) generates uncorrelated features as linear combinations of existing features in a *unsupervised* manner.

Partial Least Squares (**PLS**) **Regression** generates uncorrelated features in a *supervised* manner, i.e. by taking the relationship with the response into account.

Engineering of Categorical and Numerical Features

Outlier Correction

> An outlier is a datapoint that deviates so much from other observations that one might assume a different underlying sampling mechanism. ~ Enderlein, G. (1987)

The removal of outliers must be guided by employed model and is context-dependent.

- On the one hand, the presence of an outlier can be due to errors in the data collection processes.
- On the other hand, an outlier can be very informative and valuable data point.

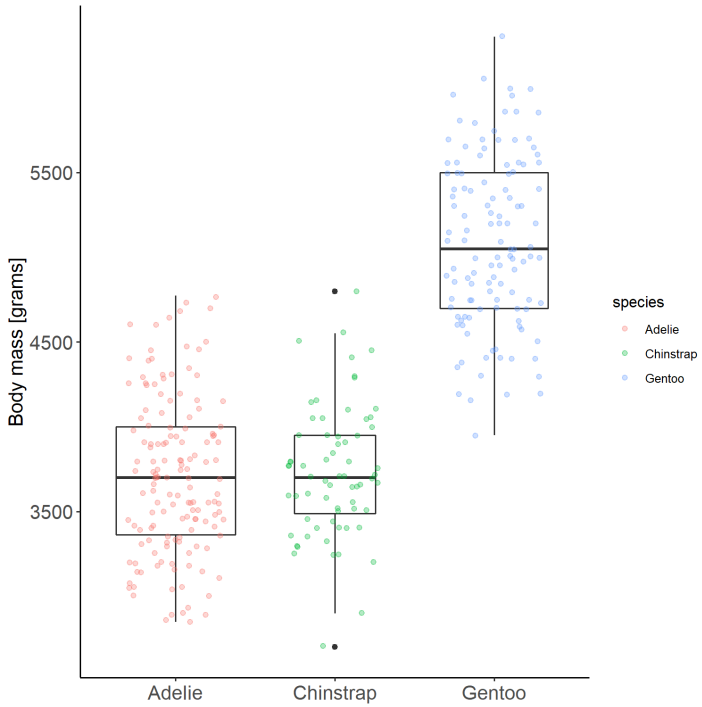
A good idea is usually to run the analysis with and without outliers and compre the results. However, in order to that, the outliers must be identified in the first place.

📈 **Visual detection techniques:** Histograms, Scatterplots, Boxplots, etc.

🧮 **Numerical detection techniques:** z-score, interquartile range, Cook's Distance

Engineering of Categorical and Numerical Features

Outlier Correction 💹

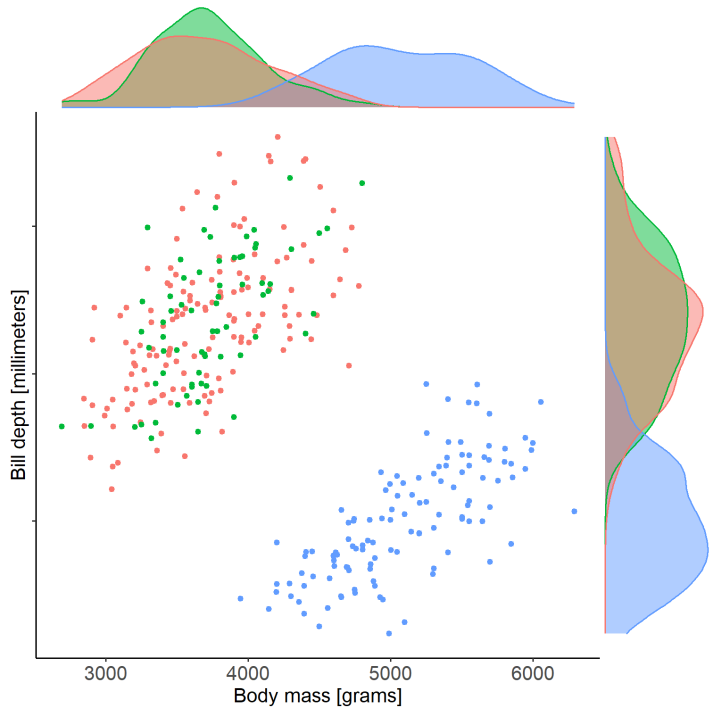


Boxplots elements:

- *Horizontal line:* Median
- *Box area:* inter-quartile range [25%-quartile; 75%-quartile]
- *Hinge length:* max. 1.5 * IQR
- *Outliers:* Points beyond the hinges

Engineering of Categorical and Numerical Features

Outlier Correction 📈



Scatterplots with marginal densities:
Convenient tool for identifying outliers in two dimensions.

- Even if a data point is quite extrem with respect to one predictor, its value with respect to other predictors can be entirely unremarkable.
- Outlier detection in higher dimensions requires techniques which take the whole feature space into account.

Note: `ggplots` with marginal densities/histograms can be generated by combining `ggplot2` with the `ggExtra` package.

Engineering of Categorical and Numerical Features

Outlier Correction 

Numerical approaches for outlier correction usually involve some form of *truncation* or *trimming* (i.e. elimination). That is, extreme values are either capped respectively artificially lowered or entirely removed from the data set to reduce their influence on the predictive model.

Some prominent techniques for pinpointing the truncation respectively trimming threshold are:

- **z-score correction:** e.g., if z-score outside the [1.96; -1.96]-interval
- **IQR correction:** if value outside of the IQR, i.e. on the hinges of the Boxplot
- **Winsorizing:** if value exceeds/undercuts a certain percentile, e.g., 95% (similar to z-score correction)

Engineering of Categorical and Numerical Features

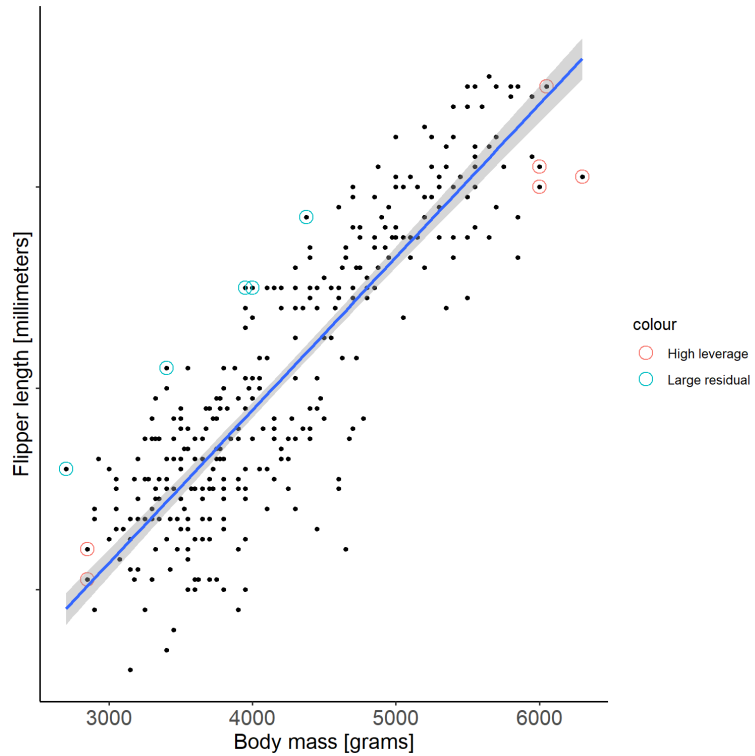
Outlier Correction

A more sophisticated, multidimensional approach goes by the name of **Cook's Distance**. It is a measure that quantifies the sensitivity of a model to the exclusion of a single observation, due to

- a large residual, i.e. a poor prediction, and
- high leverage, i.e. extreme feature values.

$$D_i = \frac{e_i^2}{p * s^2} \left[\frac{h_i i}{(1 - h_{ii})^2} \right]$$

Note: You can find multiple references in the literature with regards to the optimal cutoff threshold. Commonly, 1 is applied as a threshold for truncation respectively trimming.



Engineering of Categorical and Numerical Features

Missing Values

Some methods react highly sensitive to the presence of missing values (`NA`). That is, in certain cases the method's implementation in `R` might not even be able to fit a model in the presence of missing values.

Types of missing values:

- **Structural `NA`:** Relates to missing values that are structurally motivated (e.g., number of births per man).

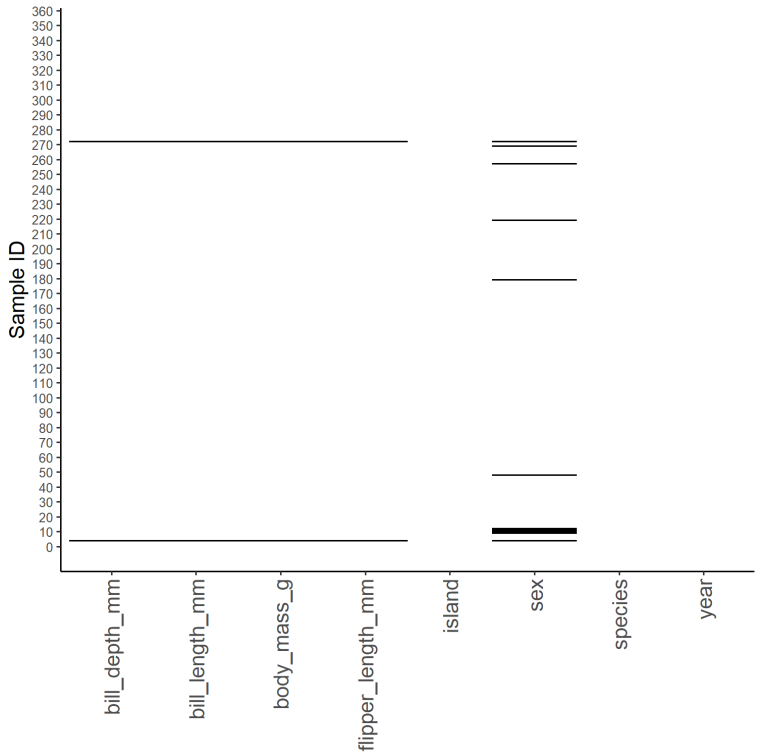
- **Random `NA`:** Relates to missing values that are due to randomness in the data generating process.
  - *Missing completely at random (MCAR):* The likelihood of `NA` is independent of the variable itself as well as any other variable in the data set. [3]
  - *Missing at random (MAR):* The likelihood of `NA` is independent of the variable itself, but can be explained by other variables in the data set. [3]

- **Non-random `NA`:** The likelihood of `NA` depends only on the variable itself (e.g., negative customer reviews). This type of `NA` is considered informative and referred to as *not missing at random (NMAR)*. [3]

Engineering of Categorical and Numerical Features

Missing Value Detection

Missing value heatmap: 📈



Check number of `NA` per feature: 🧮

```
penguins %>%
  purrr::map_dfc(., ~sum(is.na(.)))
```

Filter for missing values: 🧮

```
penguins %>%
  dplyr::filter(rowSums(across(everything(), ~ is.
```

Engineering of Categorical and Numerical Features

Missing Values: Handling of Random `NA`

Removal: Discard the observation (row) or predictor (column) containing the `NA`.

- Discard the observation (row) if the data set is relatively large or `NA`s are rare.
- Discard the predictor (column) if it is highly correlated with other predictors not subject to `NA`.

Robust algorithms: Use algorithms that are robust against the presence of `NA`. For example, the *CART* algorithm for decision trees effectively handles missing values by enabling surrogate splits.

Engineering of Categorical and Numerical Features

Missing Values: Handling of Non-Random `NA`

Mean-/median-imputation: Impute `NA` using central tendency measures, e.g., the mean or median.

Model-based imputation: Impute `NA` using an adaptive model (i.e. *imputation learner*) that predicts the missing value using all other non-missing predictors.

- The choice of the imputation learner is independent of the main prediction algorithm. You may even use a flexible imputation learner, if the final prediction model is only a simple linear regression.
- Since an individual imputation learner is required for each `NA`, the model should be efficiently implementable (e.g., kNN or trees).

`NA` as categorical predictor: Create a binary feature that indicates the presence of a `NA`. This way it can be assessed whether the `NA`-dummy is predictive of the response. However, this approach will not lead to any explanation of why this relationship may be informative as the true data is missing in the first place.

Data Leakage

Techniques for feature transformation (e.g., z-normalization), outlier correction (e.g., winsorization) or nadling missing values (e.g., mean-impuation) **must always be embedded in your resampling approach**!

- The definition of an outlier is always relative to observations in your data set.
- The amount of missing values varies from data set to data set.
- The value estimated via mean-imputation depends on the concrete predictor values in your data.
- The mean and standard deviation computed by z-normalization depends on your data set.

Altogether, each of these techniques relies heavily on the concrete train-test-split as part of the validation set or cross-validation approach. Hence, these approaches must be incorporated in the modeling pipeline.

A violation of this practice is called **Data leakage**:

Information from outside your training data (i.e. your test data) leak into the model training step. This may lead to over-optimistic models that perform extremely well on your test as they have already seen some of the test observations. This happens whenever you do computations over the whole data set.

Revisit online lecture 5 on resampling methods (in particular slides 17-21).
[5] and [6] also great resources for getting an intuitive understanding of data leakage.

Feature Selection

In general, there is little theory on how to best engineer your features. Thus, you may begin engineering a large set of feature candidates which you then narrow down. Ideally, you will be able to drop all features that do not contain any information for predicting the response. This approach is known as **feature selection**.

Reasons for feature selection:

- Parsimonious models (so-called *white box models*) are easier to interpret.
- A reduced set of features increases training speed and lowers the cost of data collection and storage.
- Feature selection helps to tackle the problem of *overfitting* and reduce the dimensionality.
- Certain models react highly sensitive to correlated predictors (e.g., linear regressions).
- Certain models react highly sensitive to uniformative predictors (e.g., SVM).

Approaches to feature selection:

- Intrinsic methods 📦
- Filter methods 🔎
- Wrapper methods 🔄

Feature Selection

Intrinsic Methods 📦

Intrinsic methods provide embedded feature selection functionalities. This broadly refers to all methods that have some type of regularization build-in, such as Lasso or pruned decision trees.

- Feature selection is directly linked to the objective function (e.g., RMSE or misclassification error).
- Intrinsic methods are model dependent and usually do not apply to other methods in the same way.
- Oftentimes *greedy* in their search for an optimal solution.

Feature Selection

Intrinsic Methods 📦

Intrinsic methods provide embedded feature selection functionalities. This broadly refers to all methods that have some type of regularization build-in, such as Lasso or pruned decision trees.

- Feature selection is directly linked to the objective function (e.g., RMSE or misclassification error).
- Intrinsic methods are model dependent and usually do not apply to other methods in the same way.
- Oftentimes *greedy* in their search for an optimal solution.

Greedy algorithms: Search for the best immediate solution without re-evaluating past choices. Greedy algorithms usually find *local optima* (e.g., stepwise subset selection).

Non-greedy algorithm: Search for the best overall solution by re-evaluating past choices. Non-greedy algorithms are able to find *global optima* (e.g., genetic algorithms or and simulated annealing).

Feature Selection

Filter Methods 🔎 and Wrapper Methods 🔄

Filter methods [4] discard features based on domain knowledge or statistical indicators, such as correlations, χ^2 scores, p-values or odds-ratios. These methods follow a rather *heurstic* approach.

Filter methods are applied once, model-agnostic (i.e. independent of the employed model), usually based on univariate/bivariate examinations (no interactions) and computationally cheap.

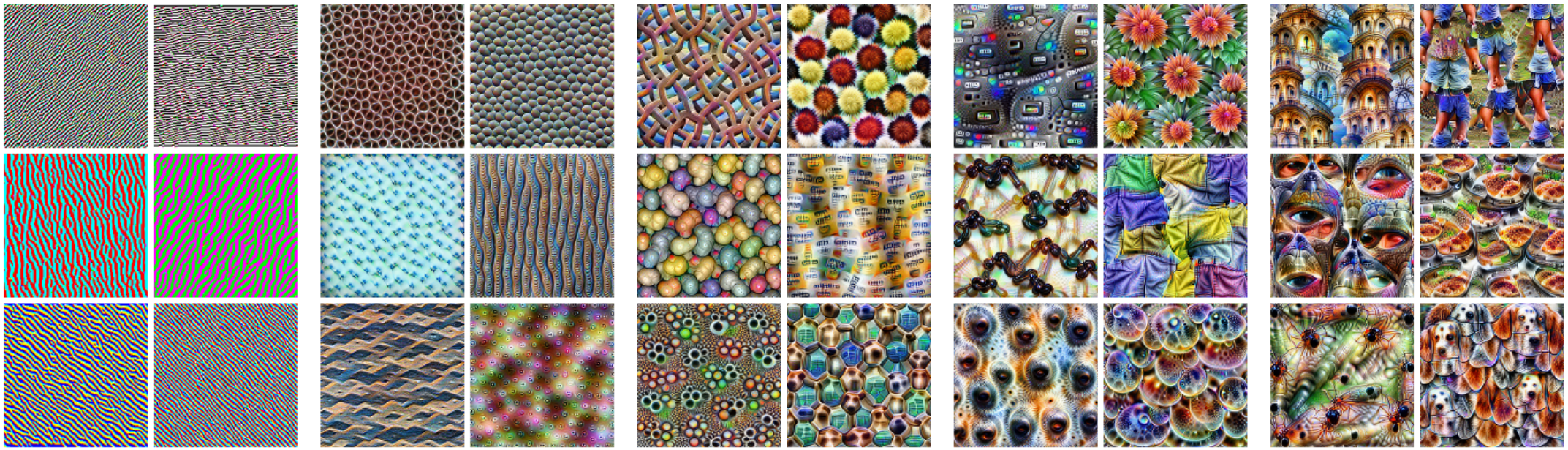
Wrapper methods [4] wrap the process of model training and feature selection into a for-loop and successively drop irrelevant features. These methods follow a rather *model-driven* approach.

Wrapper methods are iterative search procedures, based on multivariate examinations of the feature space, computationally expensive and require additional validation sets for model assessment.

Potential search strategy: (1) Start with intrinsic methods or filter methods where you are confident about the applied filters. (2) Migrate to wrappers if you have found a model specification that works for you.

Outlook

Unsupervised Feature Extraction aka Representation Learning

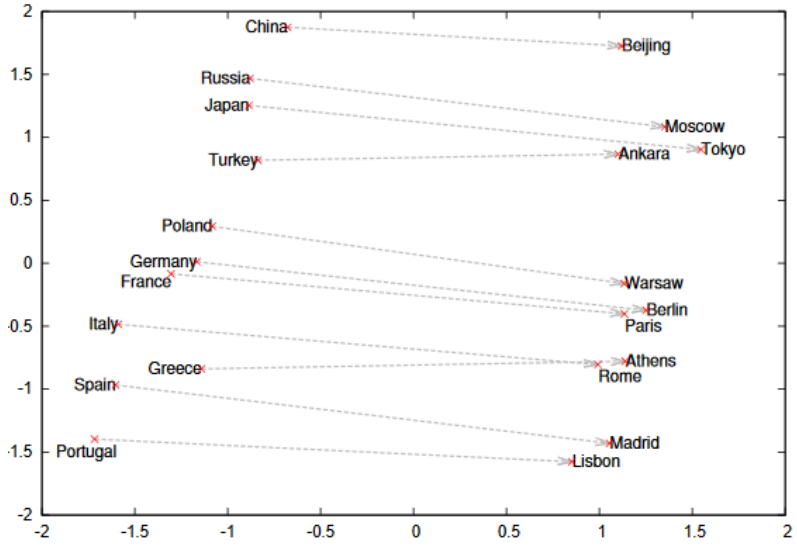


Edges (layer conv2d0) **Textures** (layer mixed3a) **Patterns** (layer mixed4a) **Parts** (layers mixed4b & mixed4c) **Objects** (layers mixed4d & mixed4e)

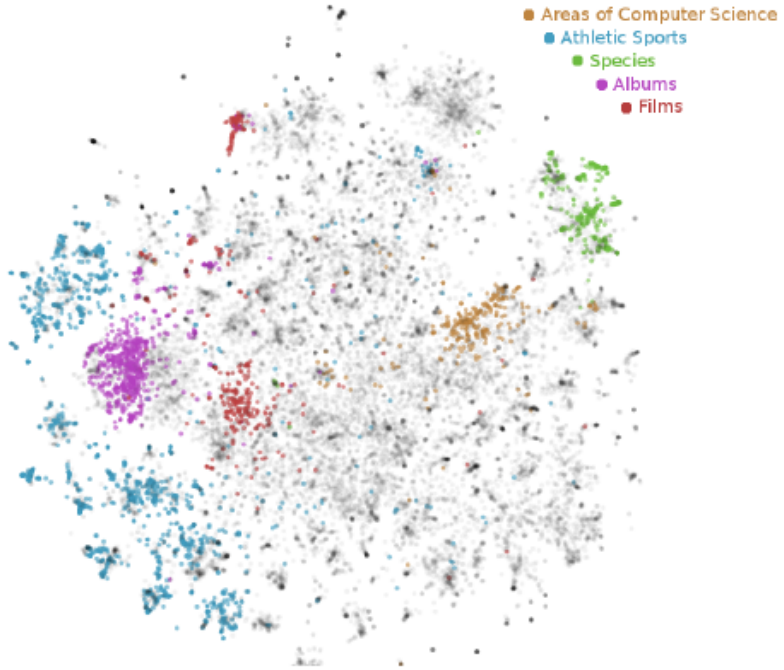
Source: [7]

Outlook

Unsupervised Feature Extraction aka Representation Learning



Source: [8]



Source: [9]

References

[1]: **Kuhn, M./Johnson, K. (2019):** Feature Engineering and Selection: A Practical Approach for Predictive Models. URL: http://www.feat.engineering/.

[2]: **Enderlein, G. (1987):** Identification of Outliers. Biometrical Journal, Vol. 29, 1987, No. 2, p. 198.

[3]: **Little, R. J./Rubin, D. B. (2019):** Statistical Analysis with Missing Data. 3rd edition. John Wiley & Sons: Hoboken 2019.

[4]: **Kohavi, R./George, H. J. (1997):** Wrappers for Feature Subset Selection. Artificial intelligence Vol. 97, 1997, No. 1-2, pp. 273-324.

[5]: **Brownlee, J. (2016):** Data Leakage in Machine Learning. Machine Learning Mastery 2016-08-02 (updated 2020-08-15). URL: https://machinelearningmastery.com/data-leakage-machine-learning/.

[6]: **Data Skeptic (2017):** [MINI] Leakage. Data Skeptic Podcast 2017-05-10. URL: https://www.youtube.com/watch?v=dI5oGNTi6Ys.

[7]: **Olah, C./Mordvintsev, A./Schubert, L. (2017):** Feature Visualization: How Neural Networks Build up Their Understanding of Images. URL: https://distill.pub/2017/feature-visualization/

References

[8]: **Mikolov, T., et al. (2013):** Distributed Representations of Words and Phrases and Their Compositionality. NIPS 2013. URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

[9]: **Dai, A. M./Olah, C./Le, Q. V (2015):** Document Embedding with Paragraph Vectors. arXiv Working Paper 2015-07-29. URL: https://arxiv.org/pdf/1507.07998.pdf.

Further Resources

Kuhn, M./Johnson, K. (2019): Feature Engineering and Selection: A Practical Approach for Predictive Models. URL: http://www.feat.engineering/.

Kuhn, M./Johnson, K. (2013): Applied Predictive Modeling. Chapter 3 (Data Pre-processing). Springer: New York 2013.

Hastie, T./Tibshirani, R./Friedman, J. (2017): The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Chapter 9.6 (Missing Data). 2nd edition, 12th printing with corrections. Springer: 2017. URL: https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12.pdf.