PYTHON PREDICTIONS

A TOBANIA COMPANY

COBRA

Predictive models focused on performance and interpretation

A recording of this presentation can be found here.

Sander Vanden Hautte & Jano Roelandt

Oct 2022

Slides inspired by

- *the 34th Data Science Leuven Meetup talk of our former colleague Jan Beníšek,*
- *Geert's article on our company website,*
- *a lot of instructions added by the current Cobra development team!*

What you’ll learn

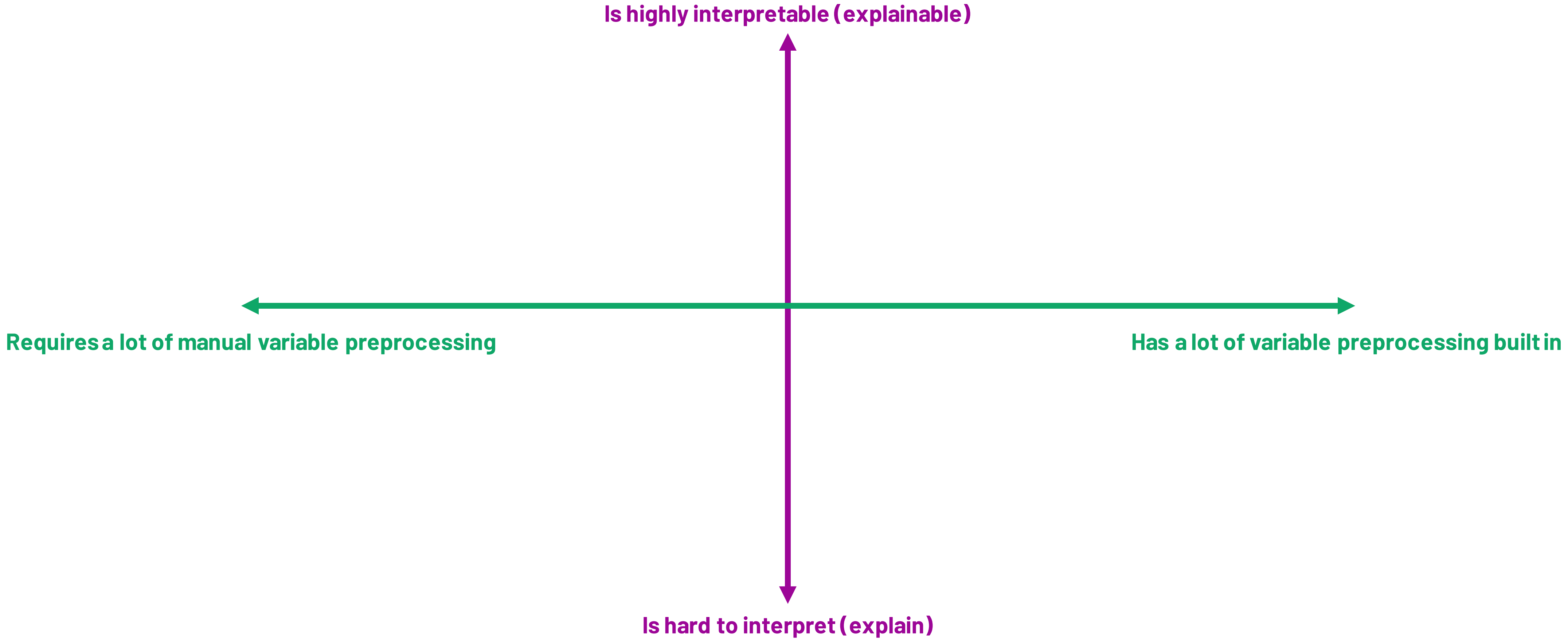
- **Data scientists:**
    - Best practices in model building
    - How to use our framework Cobra.

- **Data engineers:**
    - What Cobra serves for
    - How it is industrialized
    - How it is used in Production.

- **Data analists:**
    - Building dashboards on Cobra output
    - What the heck are your colleagues doing?

Starting exercise: where would you put the following regression/classification models?

- Deep learning (MLP/CNN/RNN)
- Decision tree
- Random forest
- Gradient boosting (XGBoost, catboost)
- Linear regression
- Logistic regression
- Cobra



Is highly interpretable (explainable)

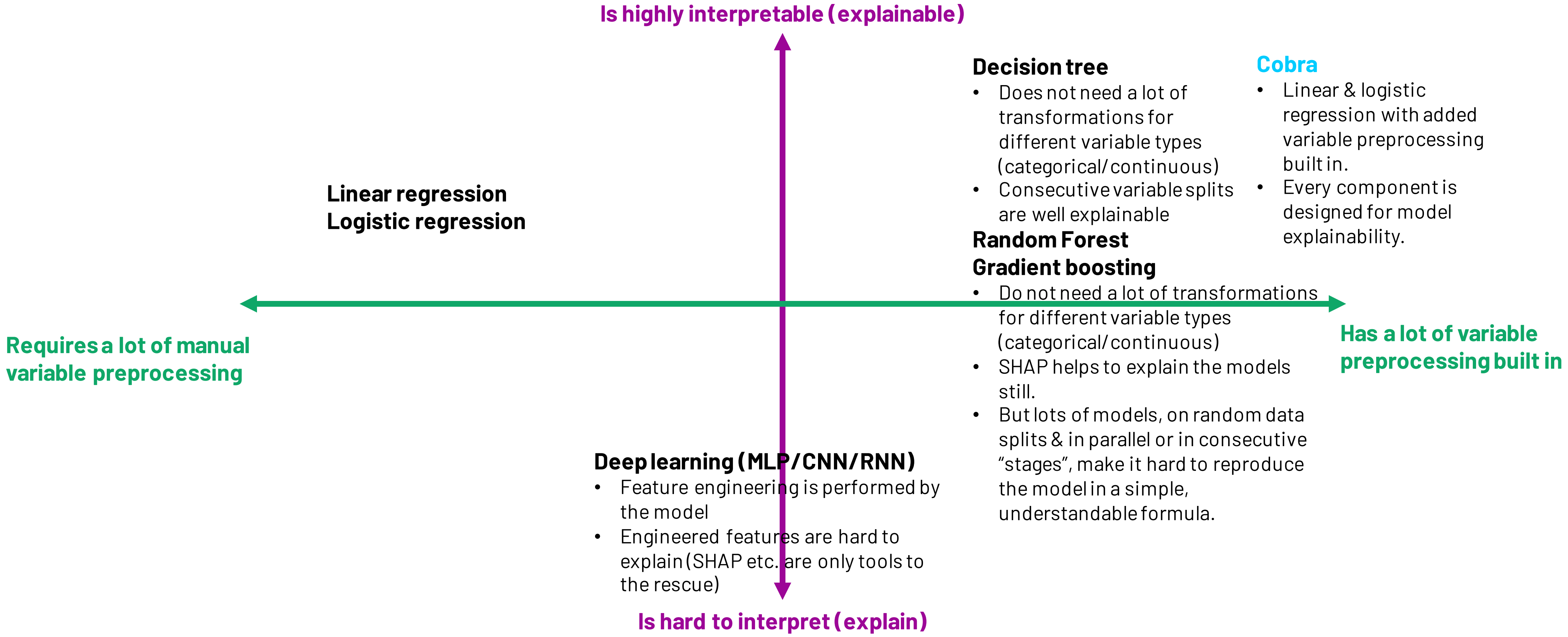
Requires a lot of manual variable preprocessing

Has a lot of variable preprocessing built in

Is hard to interpret (explain)

Starting exercise: where would you put the following regression/classification models?

- Deep learning (MLP/CNN/RNN)
- Decision tree
- Random forest
- Gradient boosting (XGBoost, catboost)
- Linear regression
- Logistic regression
- Cobra



Is highly interpretable (explainable)
Is hard to interpret (explain)
Requires a lot of manual variable preprocessing
Has a lot of variable preprocessing built in

Linear regression
Logistic regression

Decision tree

- Does not need a lot of transformations for different variable types (categorical/continuous)
- Consecutive variable splits are well explainable

Cobra

- Linear & logistic regression with added variable preprocessing built in.
- Every component is designed for model explainability.

Random Forest
Gradient boosting

- Do not need a lot of transformations for different variable types (categorical/continuous)
- SHAP helps to explain the models still.
- But lots of models, on random data splits & in parallel or in consecutive "stages", make it hard to reproduce the model in a simple, understandable formula.

Deep learning (MLP/CNN/RNN)

- Feature engineering is performed by the model
- Engineered features are hard to explain (SHAP etc. are only tools to the rescue)

WHAT?

Cobra allows you to **quickly** build **interpretable** models with high **predictive power**.

Why Cobra?

Explainability

Many organizations want to **trust a model** before putting it in production.

They prefer a **simple, interpretable model** over a complex model [1]

Coverage

Many projects are **regression or classification**: churn/acquisition/sales prediction/...

Automation

A proven approach for interpretable models, built from over 15+ years of experience at clients

Don’t repeat the approach from the ground up for every new client

Quick model building leaves more time for exploration & improvements

[1] **Of course, a simple model is *always* to be preferred over a complex one: (Ockham’s razor)**

- From a data science theoretical perspective: to reduce the risk of overfitting.
- From a resources perspective: why run a complex model, requiring more resources, if a simple model generates the same result?

Why open-sourcing Cobra?

Availability

Rapid and robust model building becomes available for many clients

Community

Sharing is caring

Giving back is important to us

Growing together

Transparency

Towards clients

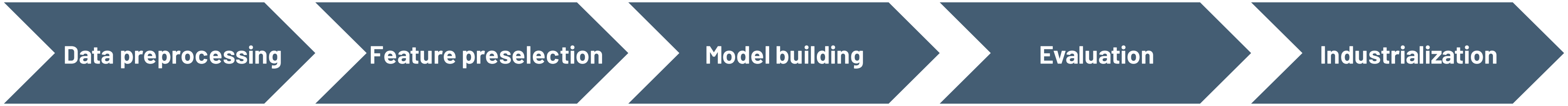
Job candidates understand what we do

And creative as we were, since "python" is in our company name, we named the framework ***Cobra*** *after yet another snake.*

HOW?

Fitting regression-based models on top of discretized predictors.

How does Cobra work?



Data preprocessing

Feature preselection

Model building

Evaluation

Industrialization

These steps are no different than any data science project.

But in each step, Cobra is developed specifically for interpretability. **(explainability by design)**

Data preprocessing

- Train/selection/validation split
- Binning continuous variables into intervals
- Imputing missing values with a category of their own (missing values bin)
- Regroup categorical variables
- Target encoder of bins/categories
- Interesting output: Predictor Insight Graphs (PIGs)

Watch out! Cobra terminology:

- Train → train
- Validation → selection (Cobra's model optimization relies most on forward feature selection)
- Test → validation (of model performance on unseen data)

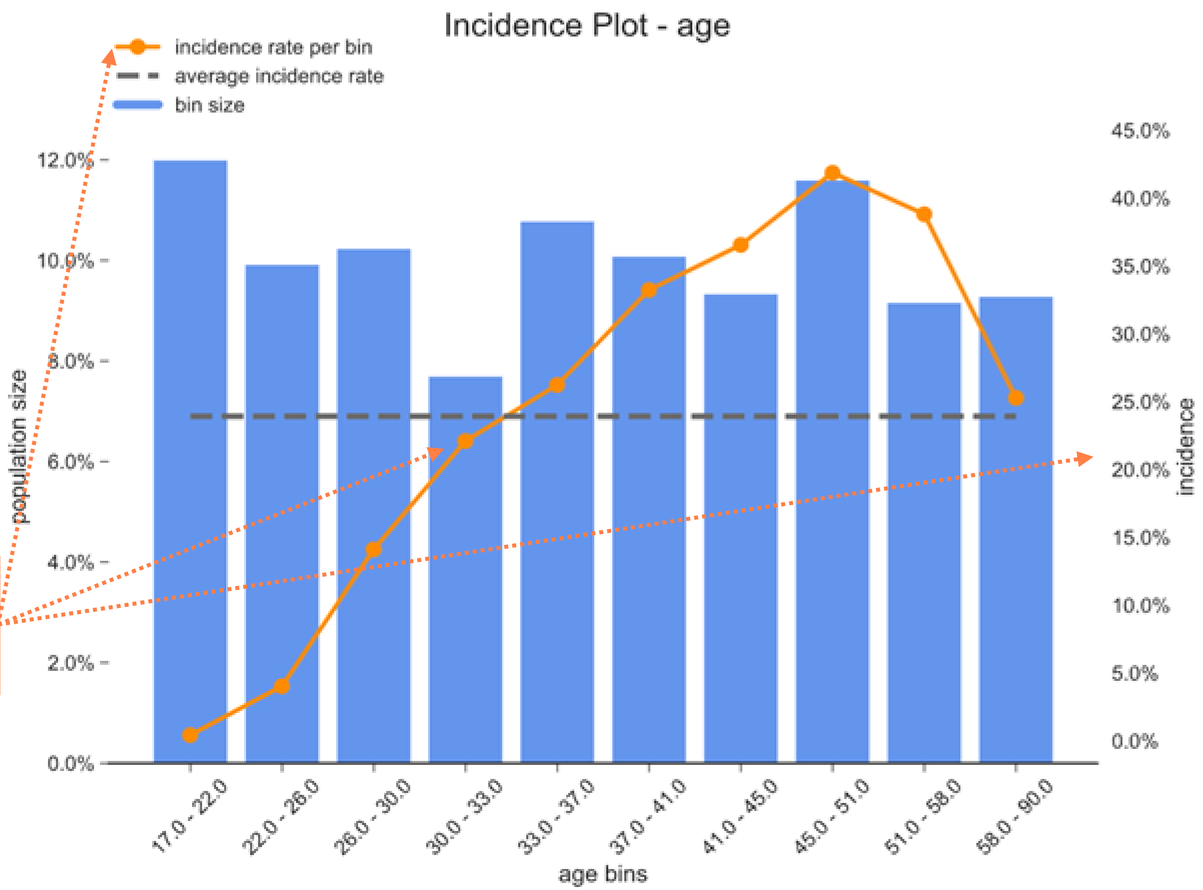
Data preprocessing
Feature preselection
Model building
Evaluation
Industrialization

Data preprocessing: PIGs

- **Interesting output: Predictor Insight Graphs (PIGs)**
  - Visualizes relationship between a single predictor and the target variable.
  - Univariate relationship with the target vs. multi-variate relationship of the model!
  - Helps discussing the predictors with the client.

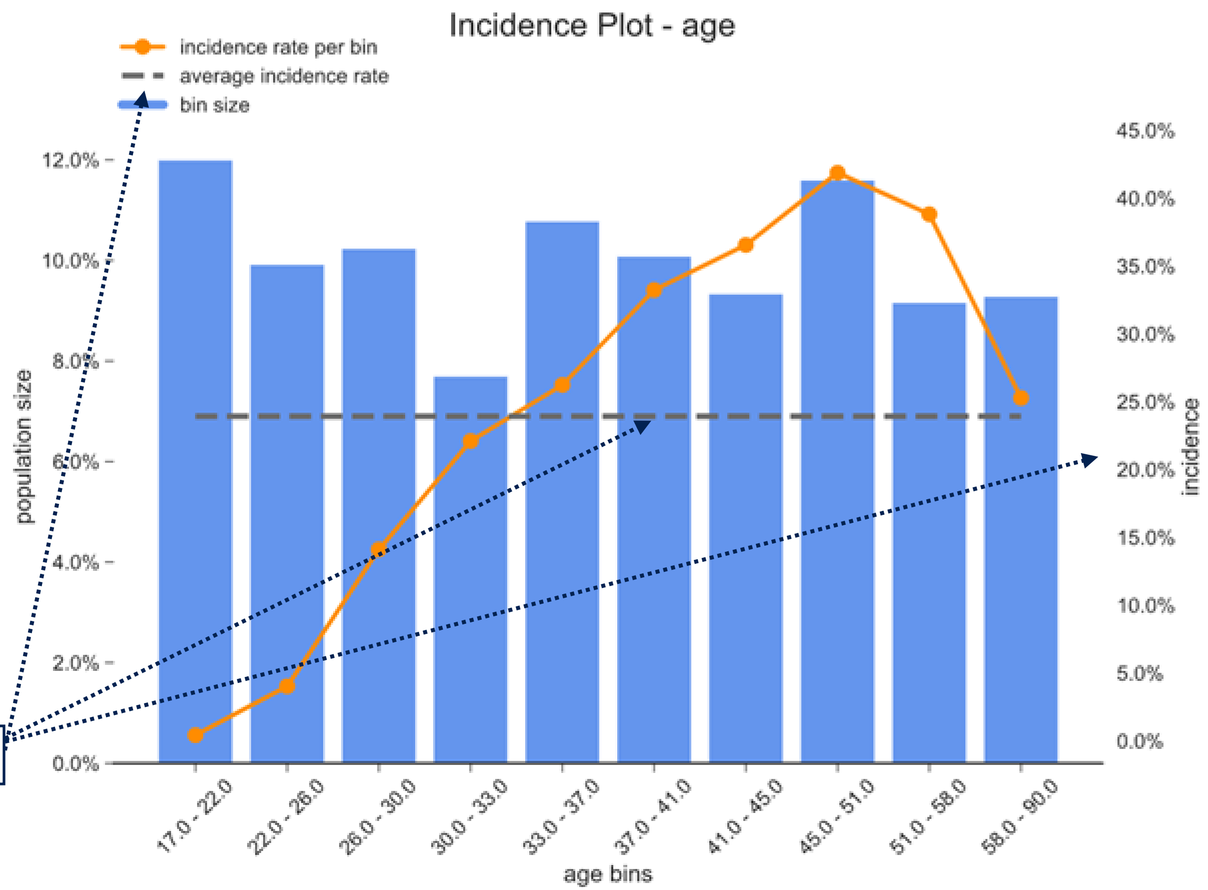
Incidence rate: the probability of the target class being 1, per bin.

In *regression* problems, the average target value per bin.



Data preprocessing: PIGs

- **Interesting output: Predictor Insight Graphs (PIGs)** 🐷
  - Visualizes relationship between a single predictor and the target variable.
  - Univariate relationship with the target vs. multi-variate relationship of the model!
  - Helps discussing the predictors with the client.

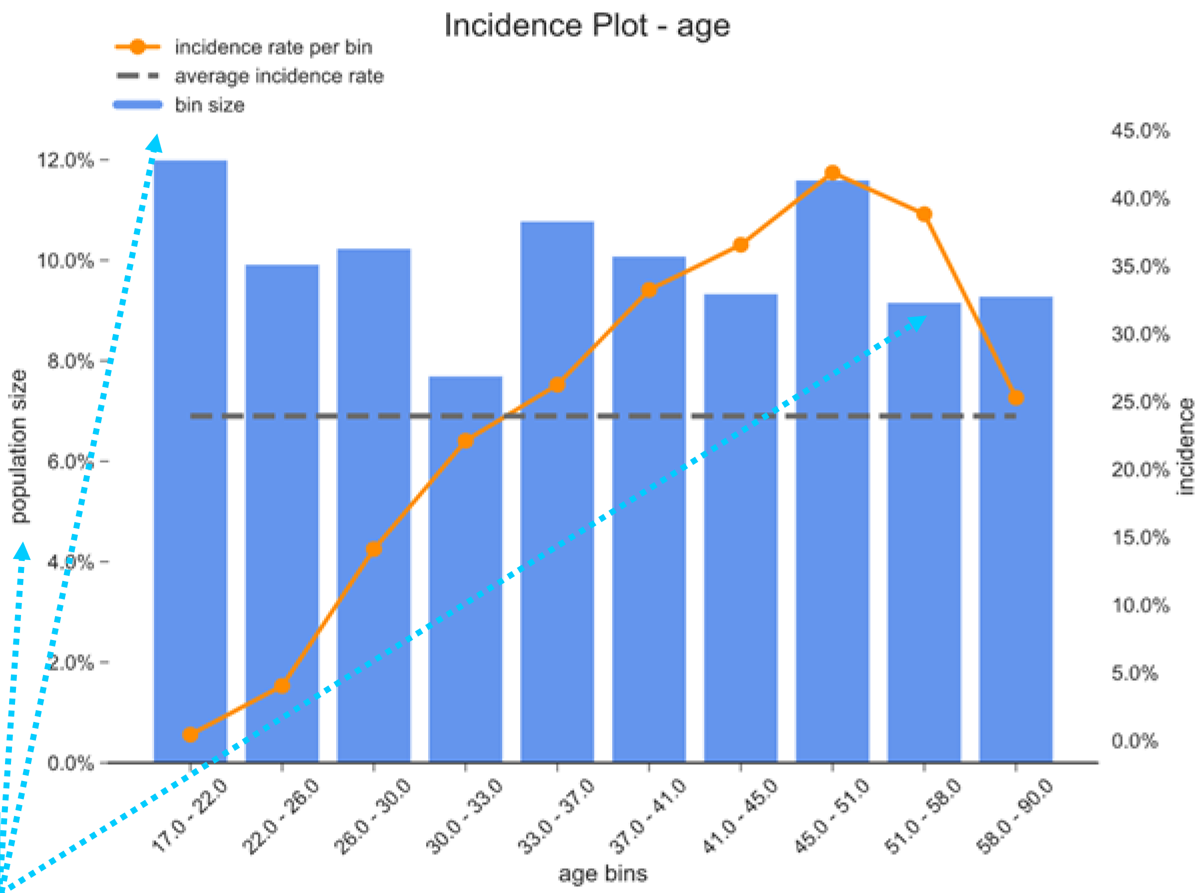


The average incidence rate: is the incidence across all bins in the dataset.

Data preprocessing → Feature preselection → Model building → Evaluation → Industrialization

Data preprocessing: PIGs

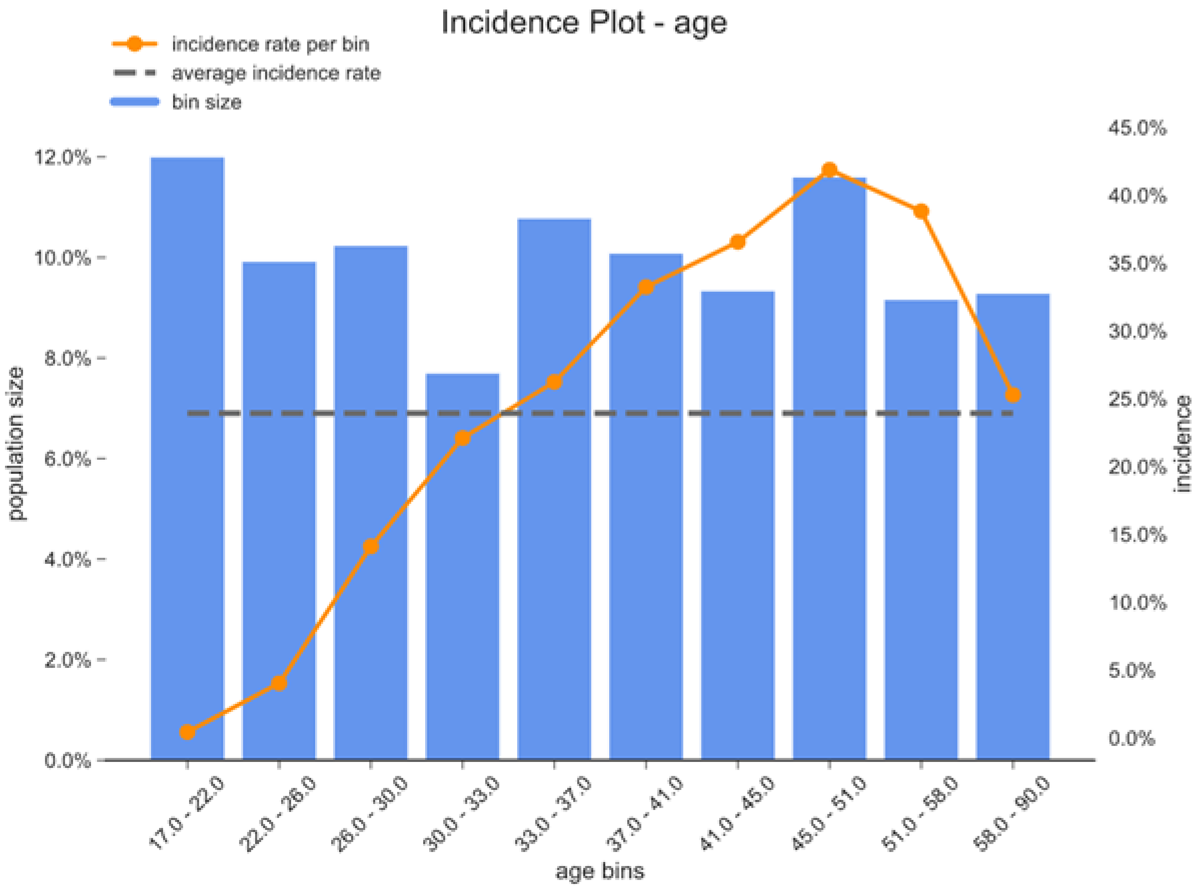
- **Interesting output: Predictor Insight Graphs (PIGs)**
  - Visualizes relationship between a single predictor and the target variable.
  - Univariate relationship with the target vs. multi-variate relationship of the model!
  - Helps discussing the predictors with the client.



The bin size: the number of samples for which the plotted variable has a value in this bin.
Verify if the number of samples in the bin is large enough to reliably inspect incidences from!

Data preprocessing: PIGs

- **Interesting output: Predictor Insight Graphs (PIGs)**
  - Visualizes relationship between a single predictor and the target variable.
  - Univariate relationship with the target vs. multi-variate relationship of the model!
  - Helps discussing the predictors with the client.



For this burnout prediction, with the PIGs for the age variable:

- *Which age group has the smallest/largest burnout risk?*
- *How much more/less risk for burnout than average over the entire population?*

Data preprocessing
Feature preselection
Model building
Evaluation
Industrialization

Data preprocessing: input data types

Cobra expects you to specify whether variables are *continous* or *categorical.*

- **Continuous variables:** age, produced amount of products, liters of motor oil,...
- **Categorical variables:**
  - Most of the time: **nominal variables** (e.g. T-shirt size S/M/L/XL/XXL)
  - Should you treat **postal code** as a continuous variable or a discrete variable?
  - Exotic case: already discretized variables.

Data preprocessing: target encoding

Better known alternatives you may have been taught in courses:

- **Label encoding**
  - For ordinal variables *(T-shirt size S/M/L/XL)*
- **One-hot encoding**
  - For non-ordinal variables *(wall paint/door paint/outdoor paint)*
  - Many variables/values can quickly result in *high memory consumption*! (why? ☺)
  - Beware of the *dummy variable trap / multicollinearity problem.* *(you only need N-1 columns)*

Last sold product ID	Paint type	Paint type – wall paint	Paint type – door paint	Paint type – outdoor paint	Target
1	Wall paint	1	0	0	1
2	Door paint	0	1	0	1
3	Outdoor paint	0	0	1	0

Data preprocessing: target encoding

- **How?**

1. **Binning:** replace the set of values of every predictor variable into a (limited number of) value bins

2. **Incidence replacement:** replace each predictor value bin with the bin's target incidence.

Last sold product ID	Customer age	Customer age – binned	Customer age – target encoded	Target
1	40	[40, 50 [	3/3 = 1.0	1
4	41	[40, 50 [	3/3 = 1.0	1
1	43	[40, 50 [	3/3 = 1.0	1
2	32	[30, 40 [	1/2 = 0.5	1
12	38	[30, 40 [	1/2 = 0.5	0
3	29	[20, 30 [	0/1 = 0	0

Data preprocessing: target encoding

- **How?**

1. **Binning:** replace the set of values of every predictor variable into a (limited number of) value bins

2. **Incidence replacement:** replace each predictor value bin with the bin's target incidence.

Back to the paint example.

Categorical variables of course are already binned.

Last sold product ID	Paint type – binned already	Paint type – target encoded	Target
1	Wall paint	3/3 = 1.0	1
4	Wall paint	3/3 = 1.0	1
1	Wall paint	3/3 = 1.0	1
2	Door paint	1/2 = 0.5	1
12	Door paint	1/2 = 0.5	0
3	Outdoor paint	0/1 = 0	0

Cobra however does some extra transformations on the bins, see next slides.

Data preprocessing: target encoding

- **How?**

1. **Binning:** replace the set of values of every predictor variable into a (limited number of) value bins

2. **Incidence replacement:** replace each predictor value bin with the bin’s target incidence.

If the target incidence (e.g. buying the product) is ***only slightly higher in post code 9210 than in 9200****, why would you keep both groups in separate bins?*

➔ *Cobra groups bins.*

Data preprocessing: target encoding

- **Advantages:**
  - **Reduced impact of outliers in continous variables**
    - Univariate variable value **outliers are added to the outside bins** (the lowest or highest bin)
    - Target is predicted with the target incidence replacement instead.
  - **Missing values are handled properly**
    - In a separate bin **(missing value bin)**
    - Customer not having provided a certain value might be important information
  - **Only modeling the relation between *interesting* nominal variable values and the target**
    - Nominal values with low occurrence are added to a **bin "Other"**
    - "Low" occurrence is configurable with a threshold and by statistical test (bin's target incidence deviates significantly from the average target incidence)

Data preprocessing: target encoding

- **Advantages:**
  - **No univariate non-linear variable transformations are needed anymore.**
    - Non-linear relationship is covered by the switch to incidences instead of the actual values.

Last sold product ID	Customer age	Customer age – binned	Customer age – target encoded	Target: Customer lifetime value
1	40	[40, 50 [	1.230.000	1.000.000
4	41	[40, 50 [	1.230.000	1.500.000
1	43	[40, 50 [	1.230.000	1.200.000
2	32	[30, 40 [	220.000	230.000
12	38	[30, 40 [	220.000	210.000
3	29	[20, 30 [	4.000	4.000

Without target encoding, a linear model would only start performing well when a new variable would be added, non-linearly transforming the customer age feature, for example **exp(customer age)**.

This requires either adding many transforms for all features or careful experimentation.

Data preprocessing: target encoding

- **Advantages:**
  - **No univariate non-linear variable transformations are needed anymore.**
    - Non-linear relationship is covered by the switch to incidences instead of the actual values.

Last sold product ID	Customer age	Customer age – binned	Customer age – target encoded	Target: Customer lifetime value
1	40	[40, 50 [	1.230.000	1.000.000
4	41	[40, 50 [	1.230.000	1.500.000
1	43	[40, 50 [	1.230.000	1.200.000
2	32	[30, 40[	220.000	230.000
12	38	[30, 40[	220.000	210.000
3	29	[20, 30 [	4.000	4.000

With target encoding, a linear model will still perform well, without the need for added non-linear transformation of features.

The model **can still learn to predict** from the other features, as well as possible, **the remaining variance on the target**, versus the target encoded customer age.

No bruteforcing of transformations or careful experimentation required.

Data preprocessing: target encoding

- So, the inclusion of target encoding in Cobra helps in **explainability by design**:
  - It helps **visualizing the predictor-target relation** in a simple way **(model explainability)**
  - Binning is rewritable as IF/ELSE statements.
    - the entire model (binning + regression formula) becomes **portable** to any execution environment.
      - Worst-case, for generating predictions in an Excel file or a SQL script.
      - Cobra models are however easily run as python objects (see last slides).

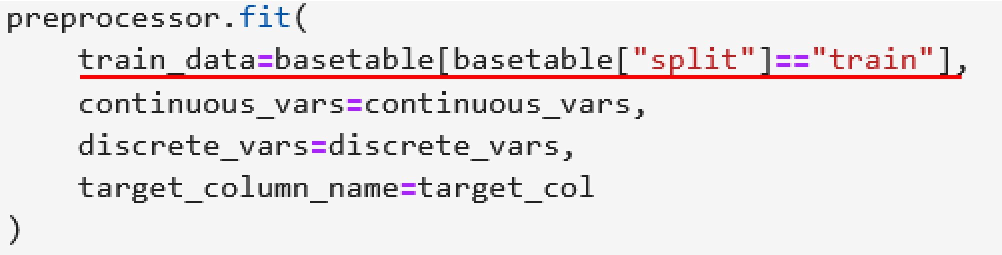
Data preprocessing: target encoding

! Important take-away!

- **!WARNING!**

  DON’T leak the target incidences from the TEST set into the preprocessing of the dataset.

  - Do **not** pass the selection or validation set here:



```
preprocessor.fit(
    train_data=basetable[basetable["split"]=="train"],
    continuous_vars=continuous_vars,
    discrete_vars=discrete_vars,
    target_column_name=target_col
)
```

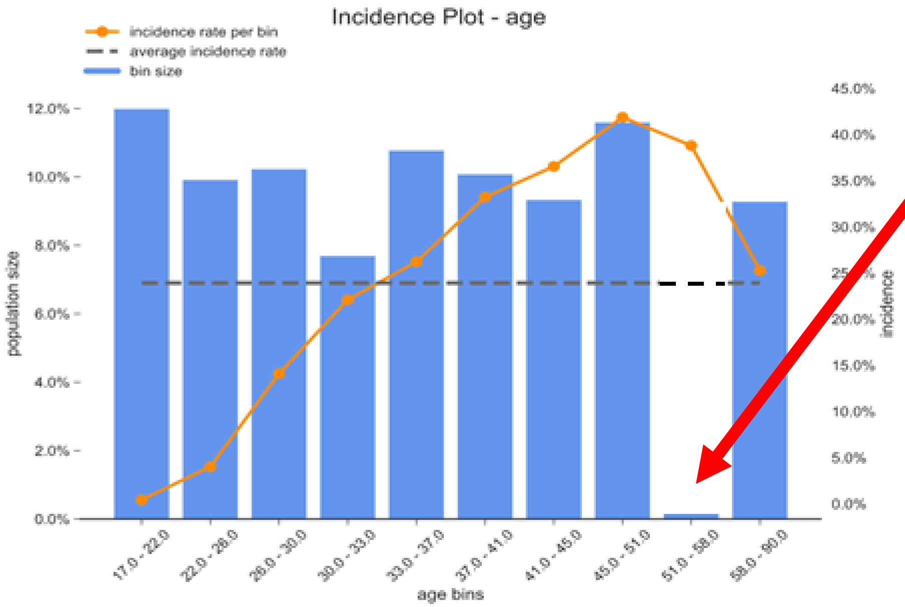
  - This information leakage will **compromise** objective validation of model generalization performance on unseen data.

Data preprocessing: target encoding

- **!WARNING! Largely different bin sizes can indicate…:**
  - **… that unimportant bins can be grouped**
    - Do not keep that *one* sample customer with an old product, as the target incidence may be a bad measure for the incidence of the bigger population with that product.
    - Group small bins together with other bins, **otherwise overfitting the model!** (bad generalization to unseen data)
    - **By default, creating 10 bins** per predictor is often sufficient to cover the non-linearity and prevent overfitting.
    - In small datasets (e.g. under 5K observations), reducing the number of bins typically reduces overfitting.” (do you understand why?) + if doubting how many bins, take the #bins required in a univariate decision tree that results in best model performance (again, often just 10).

Data preprocessing: target encoding

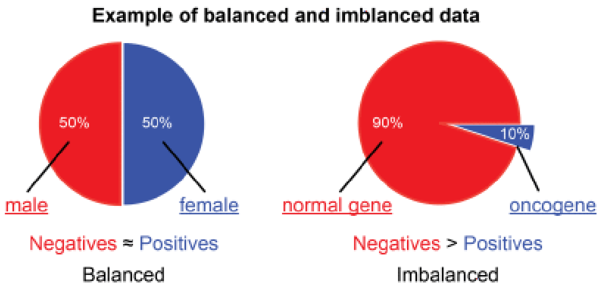
- **!WARNING! Largely different bin sizes indicate:**
    - **That you should consider grouping more bins together.**
        - Adapt Cobra's preprocessing parameter n_bins = 10
        - Or group the input column yourself before Cobra preprocessing.



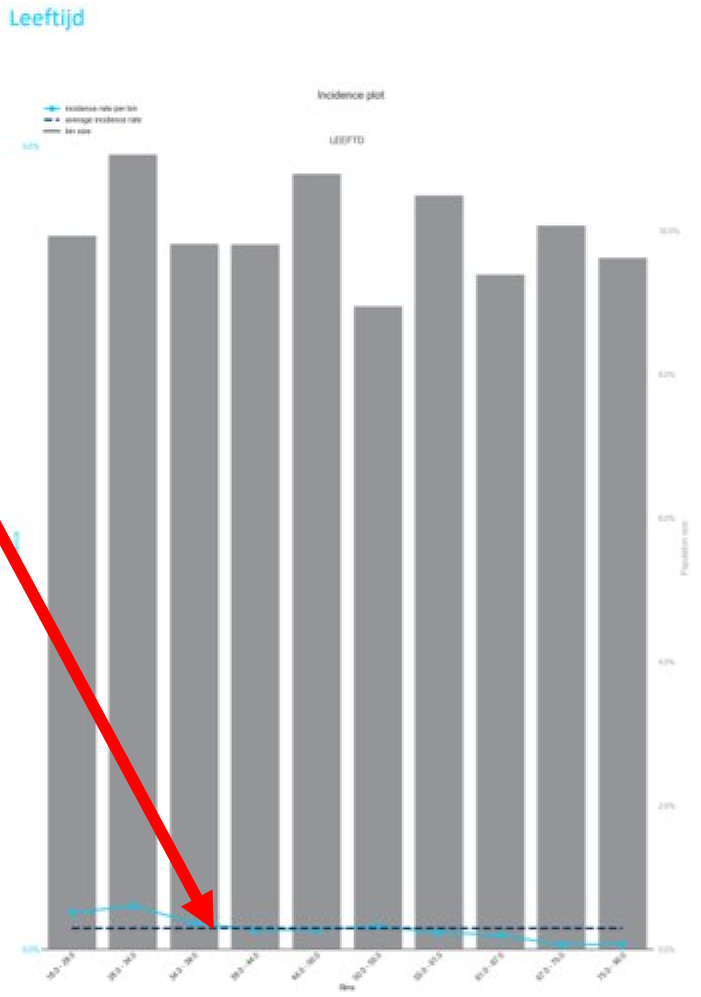
(Note: this is a contrived, artificially generated example.)

Data preprocessing: target encoding

- **!WARNING! Generally low incidence indicates:**
  - **That the dataset *target* is also very imbalanced**
    - For example 1%-99% or worse…

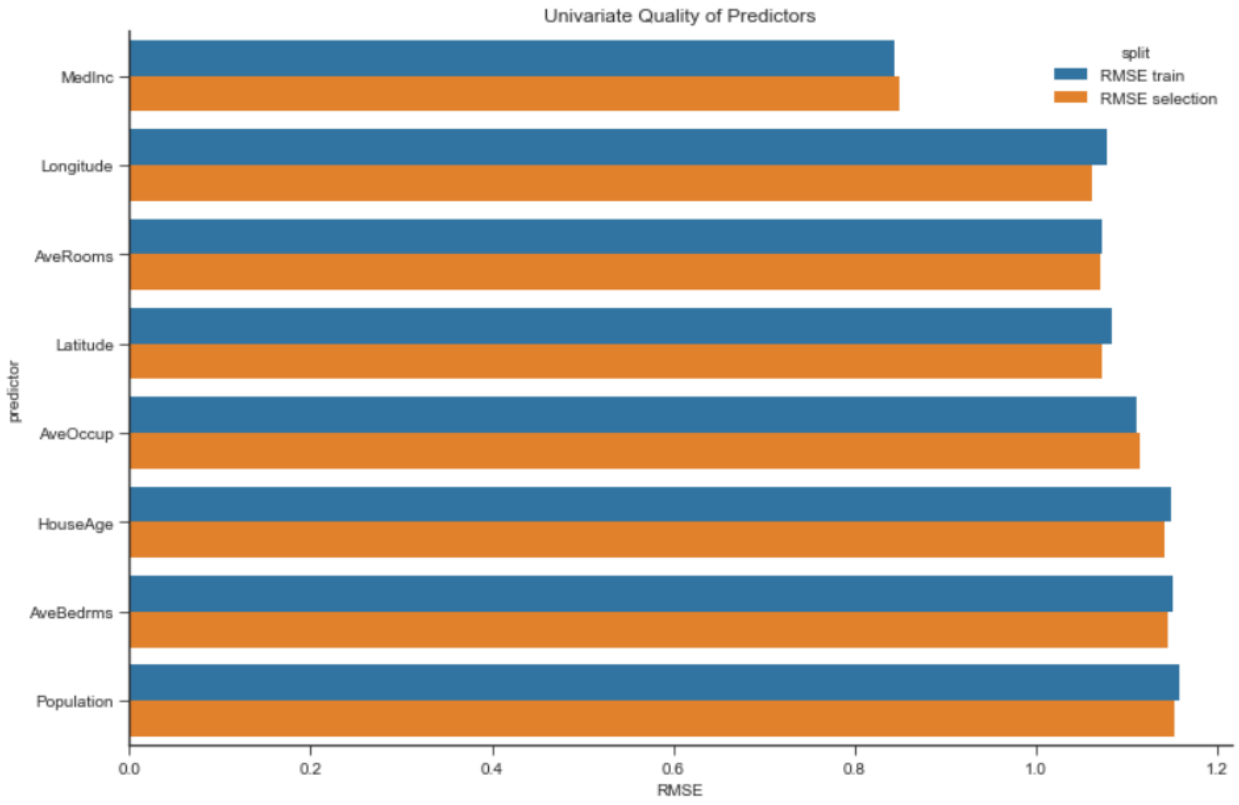


    - **Try stacking basetables**, for example for many month snapshots in the past, instead of just one.
    - **Try sample weighting** in the model (try **class_weights** hyperparameter first of all!)
    - If unsuccessful, try undersampling or oversampling (SMOTE, ADASYN, etc.).



Univariate feature pre-selection

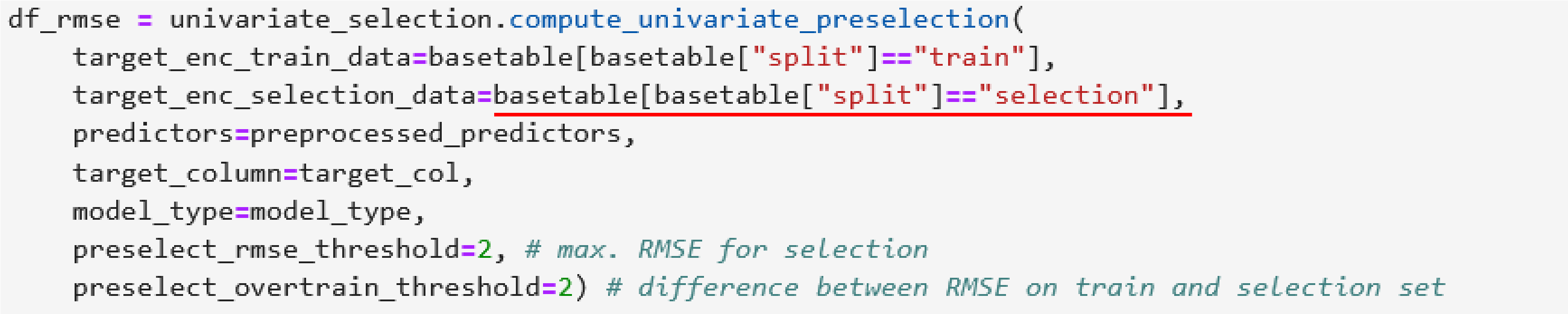
- **Univariate pre-selection:** *we select only **strong** features and drop those which **overfit**.*
  - Regression example: if one variable on its own results in a very bad model (very high RMSE), it may not be worth including.
  - May only be worth including if it helps building (non-linear) feature interactions.
- **Default evaluation metric:**
  - AUC for classification
  - RMSE for regression.



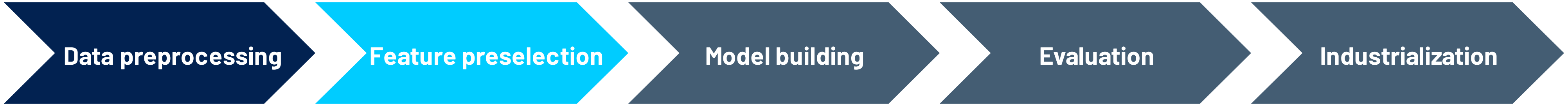
Data preprocessing
Feature preselection
Model building
Evaluation
Industrialization

Univariate feature pre-selection

- Validation on the **selection** set, not the validation set.



```
df_rmse = univariate_selection.compute_univariate_preselection(
    target_enc_train_data=basetable[basetable["split"]=="train"],
    target_enc_selection_data=basetable[basetable["split"]=="selection"],
    predictors=preprocessed_predictors,
    target_column=target_col,
    model_type=model_type,
    preselect_rmse_threshold=2, # max. RMSE for selection
    preselect_overtrain_threshold=2) # difference between RMSE on train and selection set
```



Data preprocessing

Feature preselection

Model building

Evaluation

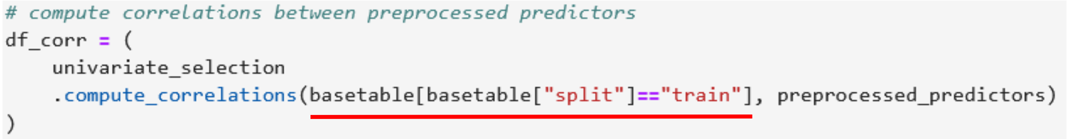
Industrialization

Feature correlations

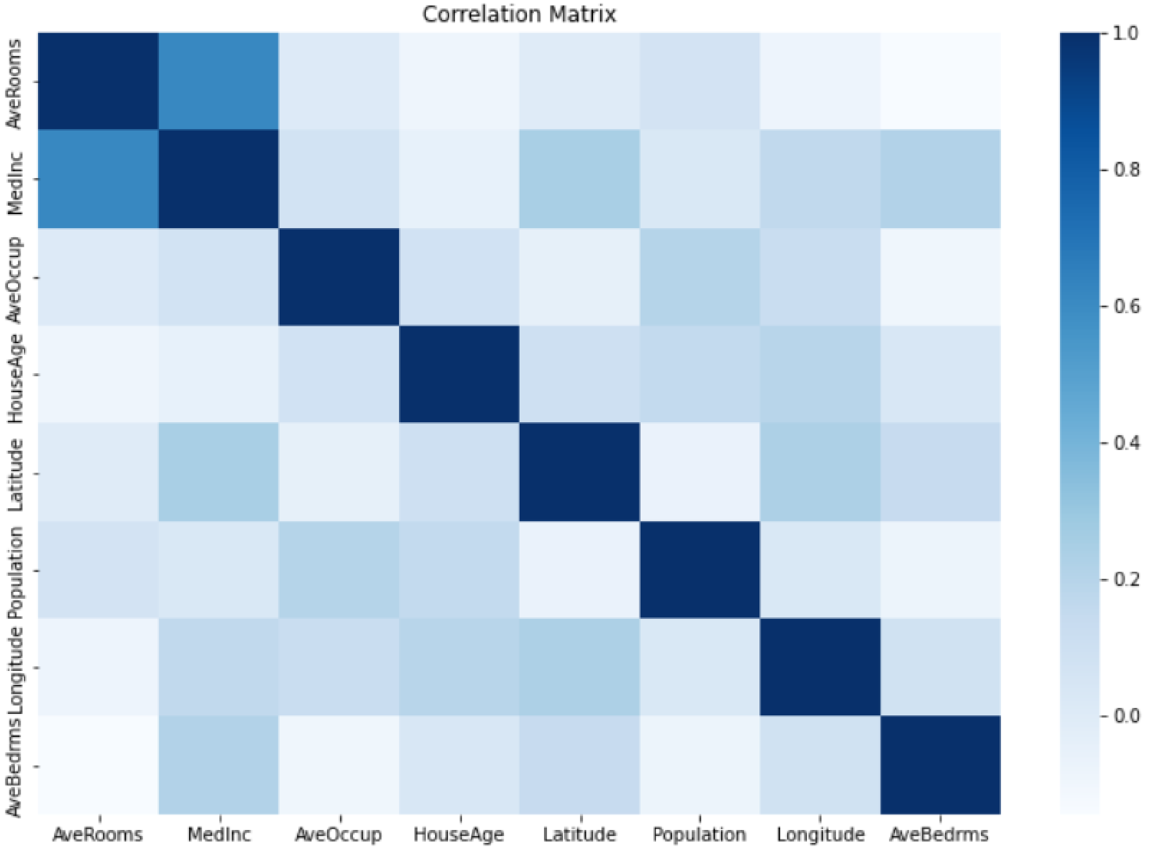
- **The customer may prefer an alternative variable to be used in modeling**
  - The variable to be replaced can be excluded (parameter excluded_predictors).

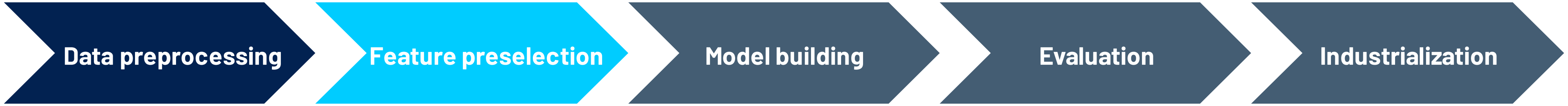
- **Correlations explain why only a few variables are required for a good model**
  - Adding more variables will not increase model performance anymore (see forward selection in next slides).

- It's best to not "peek ahead": investigate correlations preferably on the **training set**.



```
# compute correlations between preprocessed predictors
df_corr = (
    univariate_selection
    .compute_correlations(basetable[basetable["split"]=="train"], preprocessed_predictors)
)
```





Data preprocessing

Feature preselection

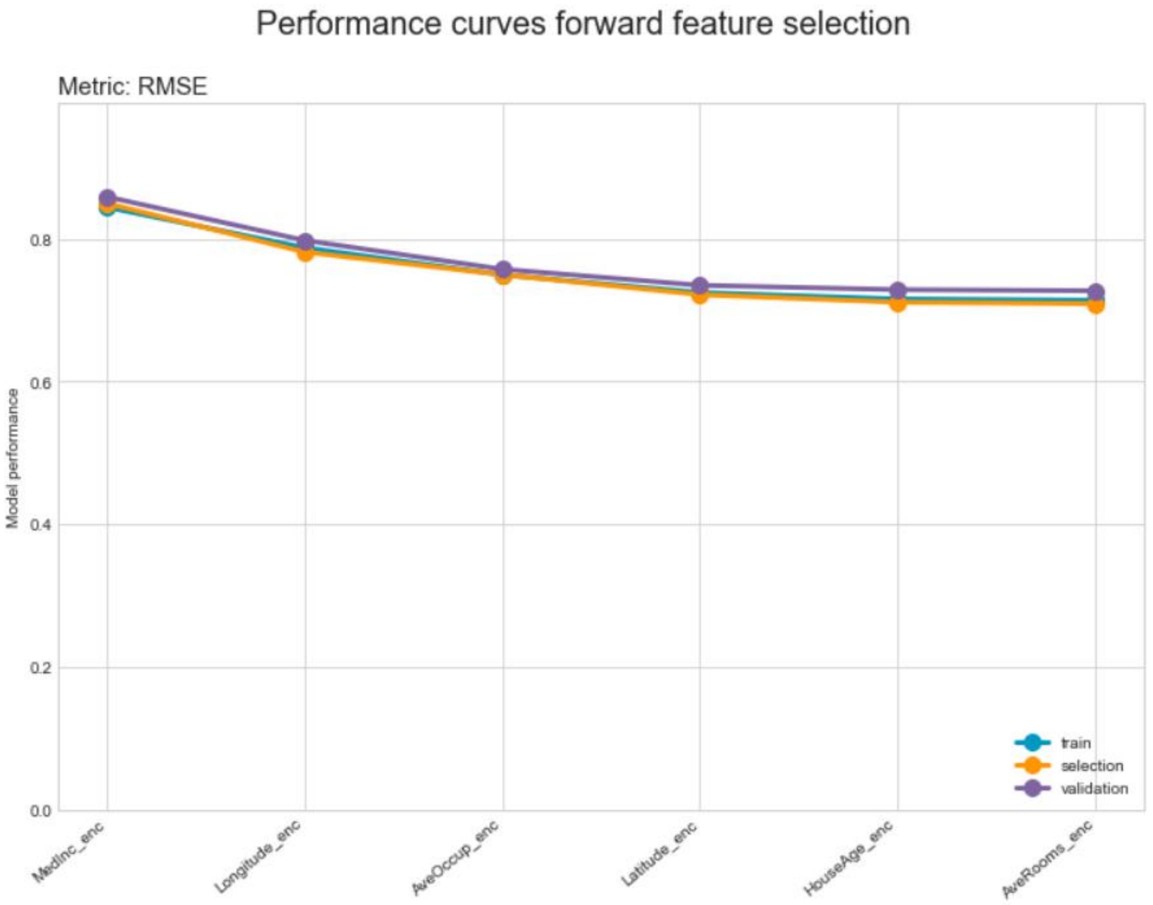
Model building

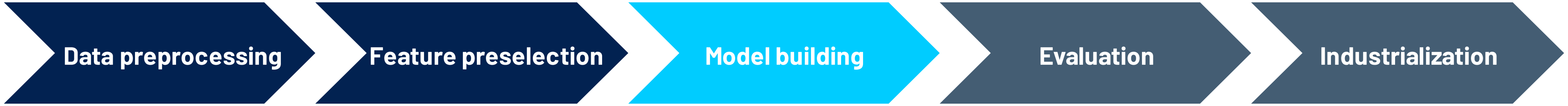
Evaluation

Industrialization

Model building: multi-variate feature selection.

- **Forward feature selection**
  - **Sequentially adding the next top-performing variable** to the model predictors.
    - Many models are built.
    - Similar to scikit-learn's SelectKBest
  - So, ***multi-variate* selection** this time!
  - **Validation on the selection set**, not the validation set.

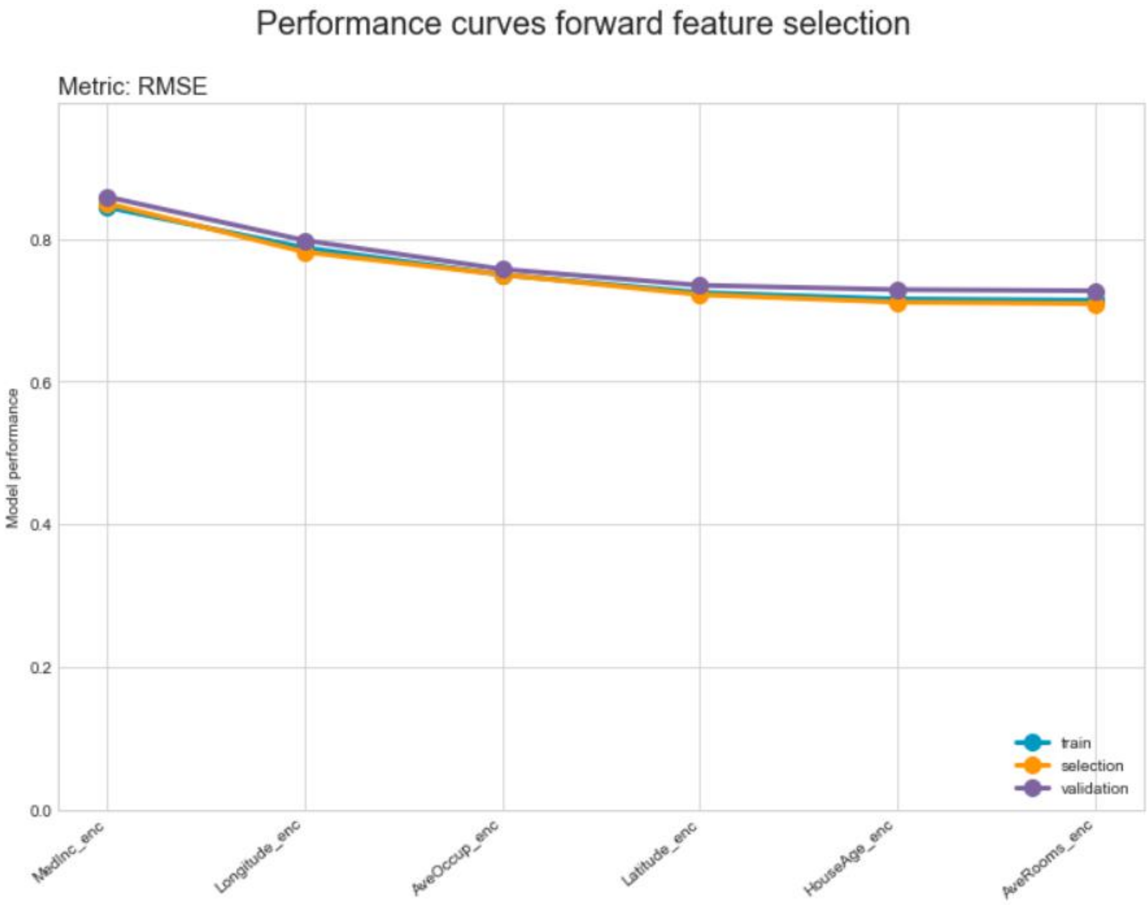




Model building: multi-variate feature selection.

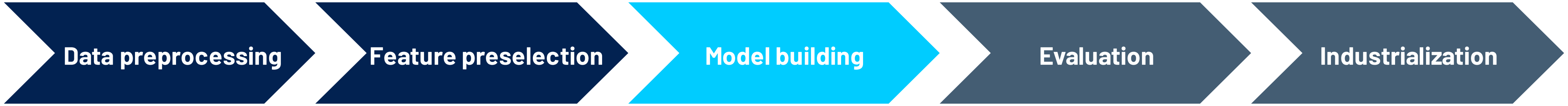
- **How many features do you select?**

- Very useful to confirm this with the client's business knowledge.
- Keep in mind: correlated features (see correlation slides).



Performance curves forward feature selection

Metric: RMSE



Data preprocessing

Feature preselection

Model building

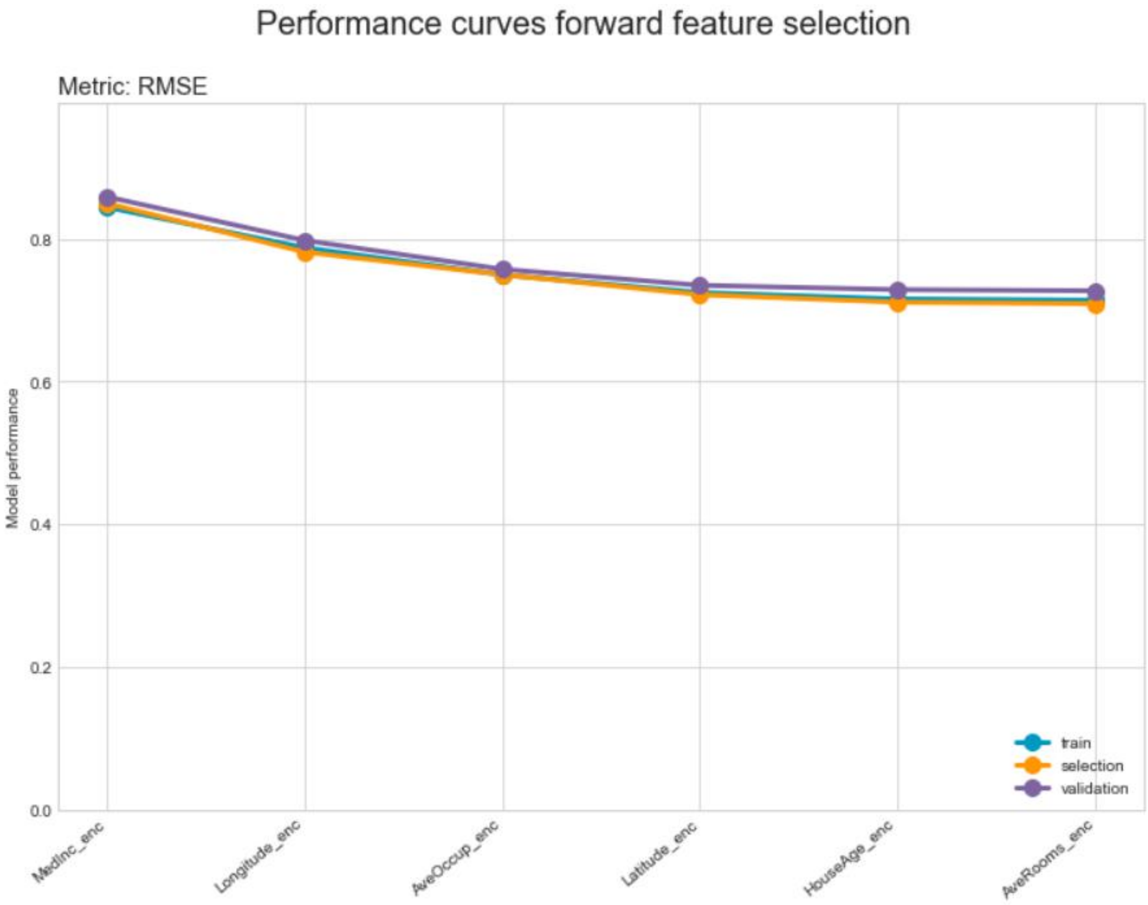
Evaluation

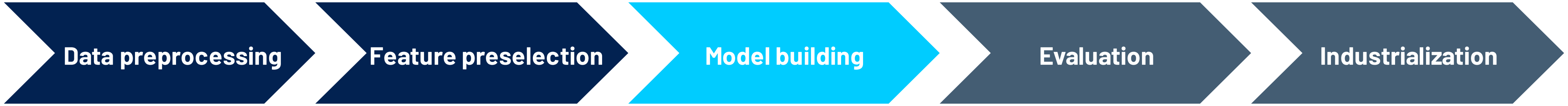
Industrialization

Model building: multi-variate feature selection.

- **Default evaluation metric:**
  - AUC for classification
  - RMSE for regression.
  - **Another one can be chosen**

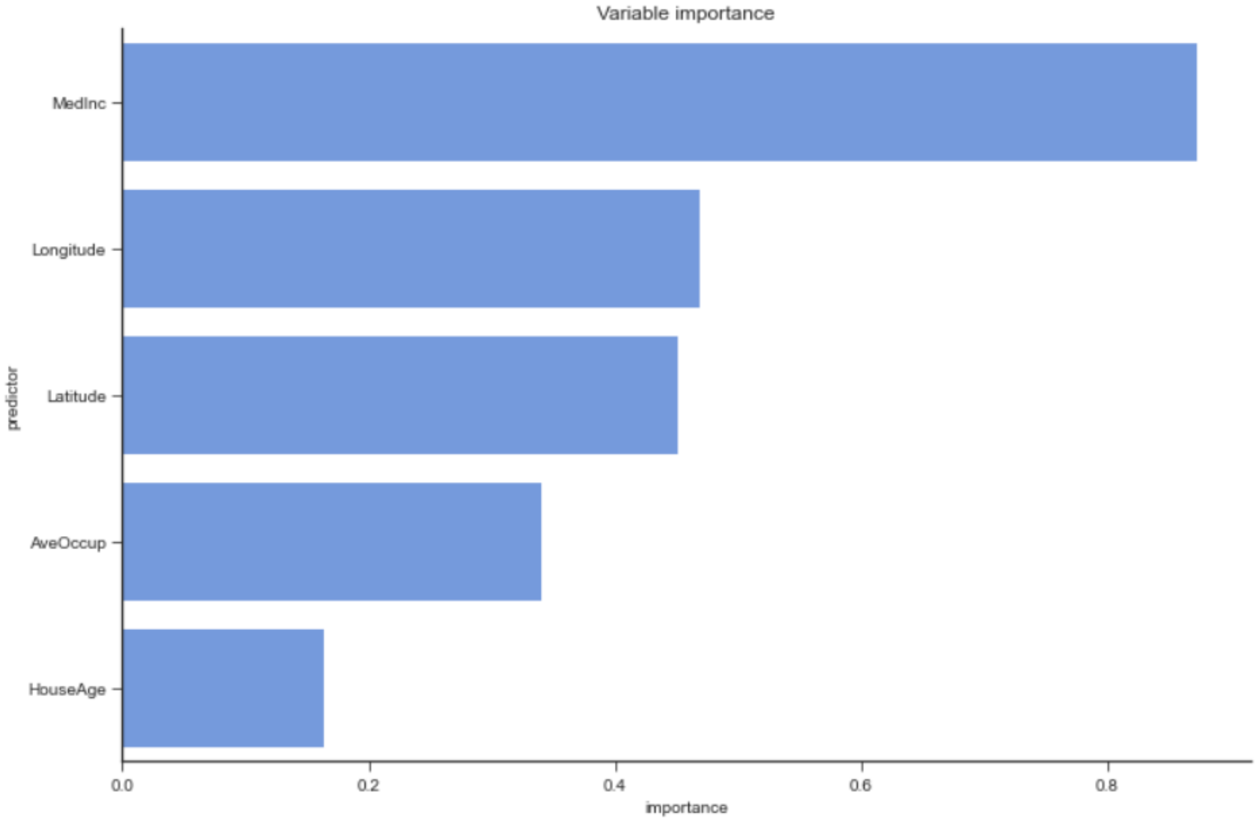
For example **lift**, when known upfront the goal is a campaign in which only the top 10% likely customers should be targeted.

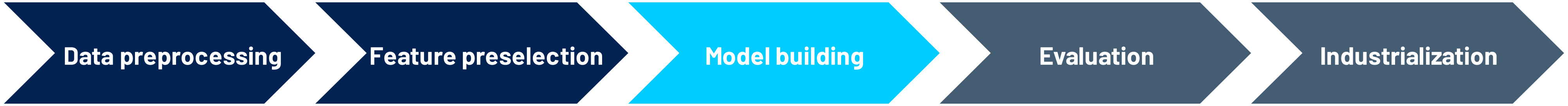



Data preprocessing
Feature preselection
Model building
Evaluation
Industrialization

Model building: Feature importance plots

- Tells the customer which factors are in play the most – can be very important.




Data preprocessing

Feature preselection

Model building

Evaluation

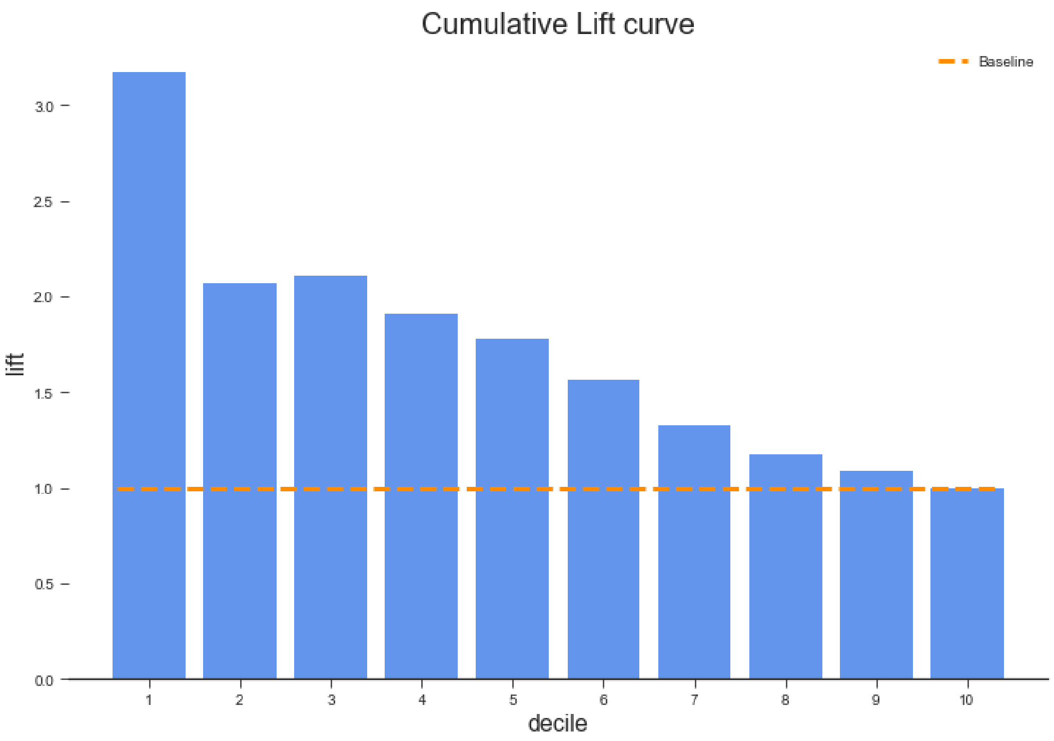
Industrialization

Model evaluation: other interesting tools

- **The PIG plots of course!**

- **Plots of ROC, Confusion matrix, lift, ...**
    - We use **lift** often for marketing campaign projects.
    - See the **cobra.evaluation** module.

- **Scalar metrics**
    - Accuracy, AUC, ... on the validation set (test set)
    - Used most often internally, not in communication to the client.



Data preprocessing

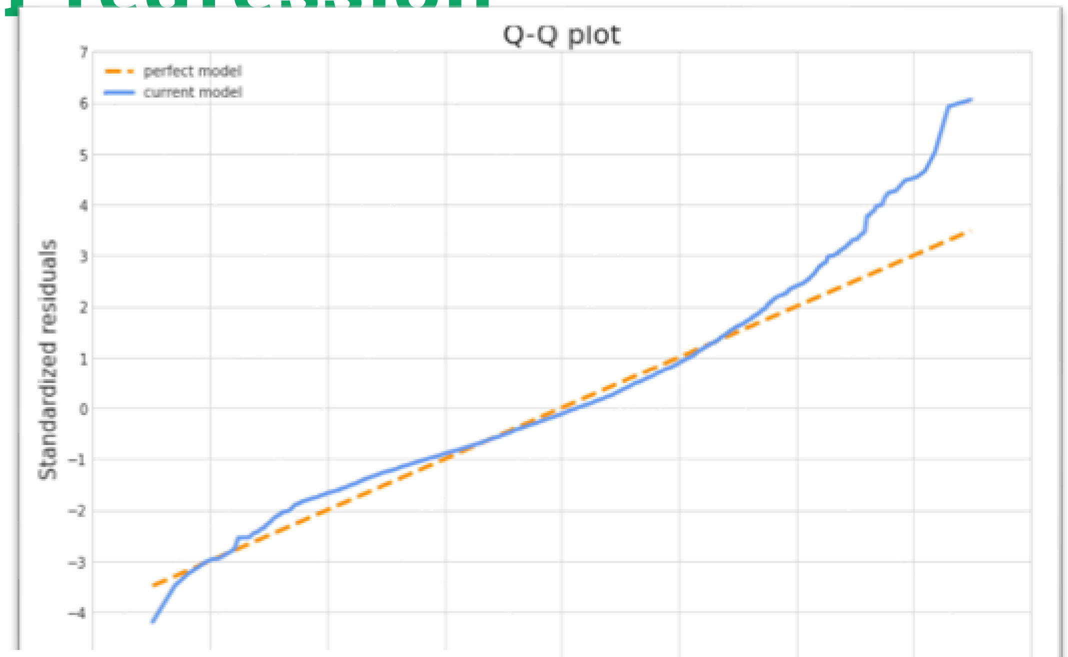
Feature preselection

Model building

Evaluation

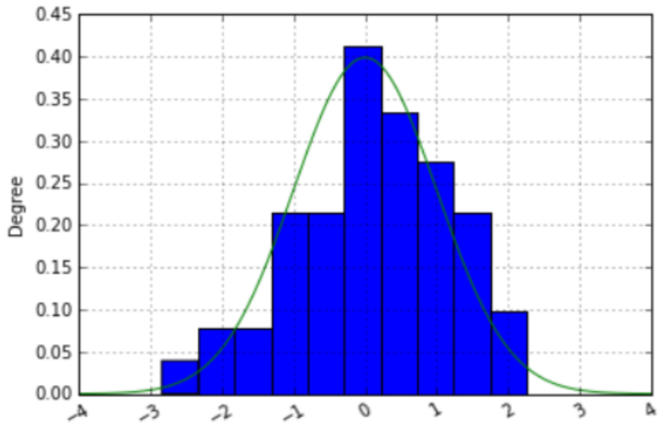
Industrialization

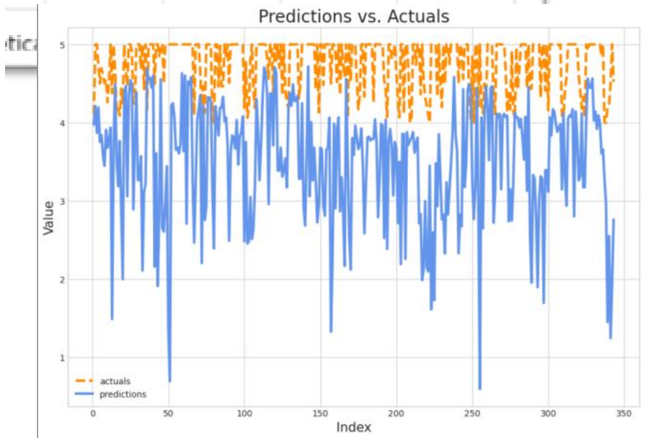
Model evaluation: Q-Q plot for regression



- **Helps you investigate the residuals (model errors)**
  - Consistent **bias** in the predictions vs. the expected value?
  - Large **variance** on the predictions vs. the expected value?
  - Is bias/variance **different for different groups of samples** in the population?

For example: high prediction variance on low and high house prices, OK bias & variance on normal house price ranges.



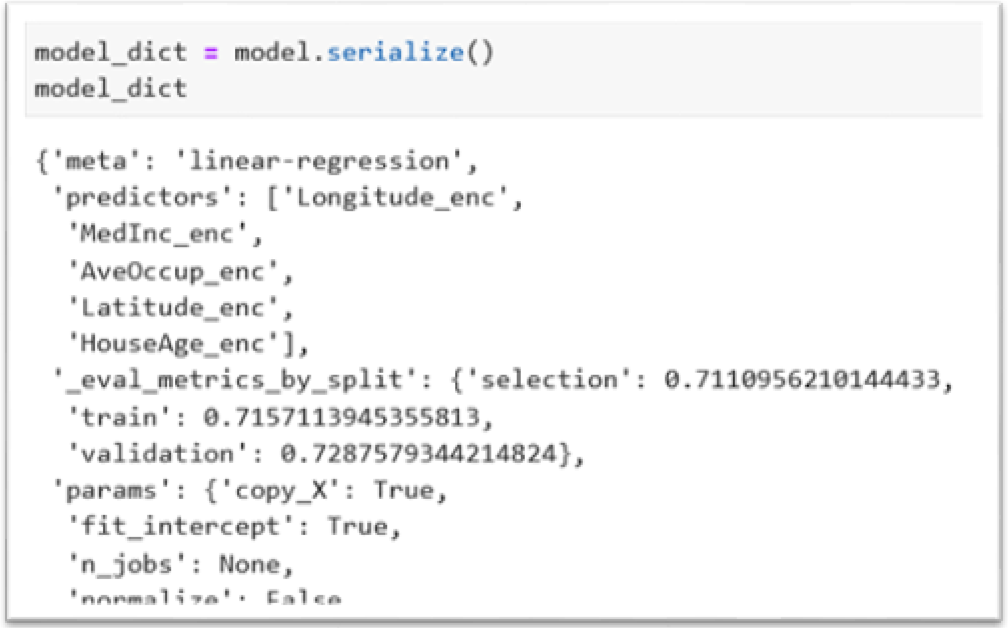


Data preprocessing
Feature preselection
Model building
Evaluation
Industrialization

Model industrialization

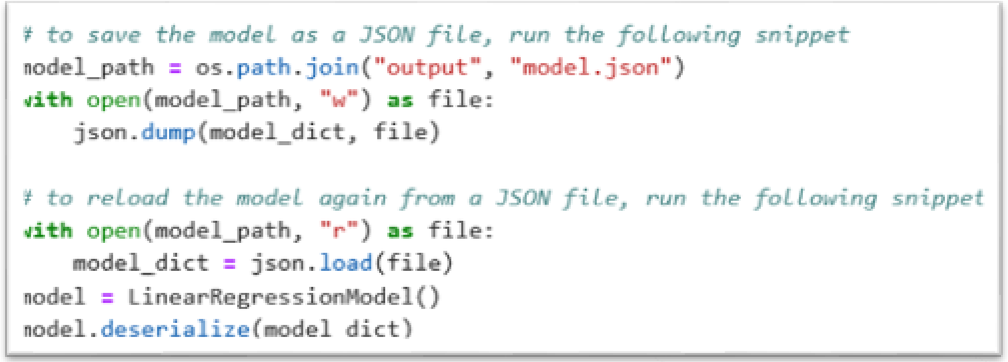
- **Configuration persisted in 2 JSON objects:**
  - The preprocessor
  - The model

- **Use cases in production:**
  - **Regular scoring**
    - Means preprocessing + model scoring
  - **Model monitoring**
    - Persisting the JSON configuration of preprocessor & model
    - Persisting the obtained model scores
  - **Re-fitting** currently requires a small change to feature selection to be automated.
    - Heuristic selection of the best number of features
    - Now requiring manual re-fit by a data scientist.
    - This feature is on our Cobra extension wishlist!



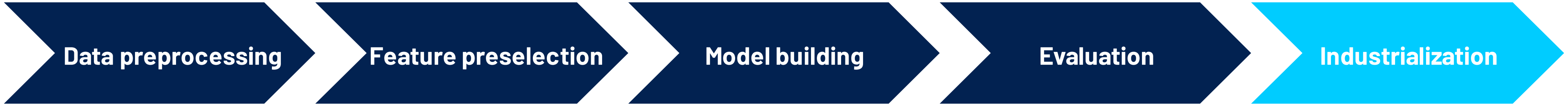
```
model_dict = model.serialize()
model_dict
```

```
{'meta': 'linear-regression',
 'predictors': ['Longitude_enc',
  'MedInc_enc',
  'AveOccup_enc',
  'Latitude_enc',
  'HouseAge_enc'],
 '_eval_metrics_by_split': {'selection': 0.7110956210144433,
  'train': 0.7157113945355813,
  'validation': 0.7287579344214824},
 'params': {'copy_X': True,
  'fit_intercept': True,
  'n_jobs': None,
  'normalize': False
```



```
# to save the model as a JSON file, run the following snippet
nodel_path = os.path.join("output", "model.json")
vith open(model_path, "w") as file:
    json.dump(model_dict, file)

# to reload the model again from a JSON file, run the following snippet
vith open(model_path, "r") as file:
    model_dict = json.load(file)
nodel = LinearRegressionModel()
nodel.deserialize(model_dict)
```



Data preprocessing

Feature preselection

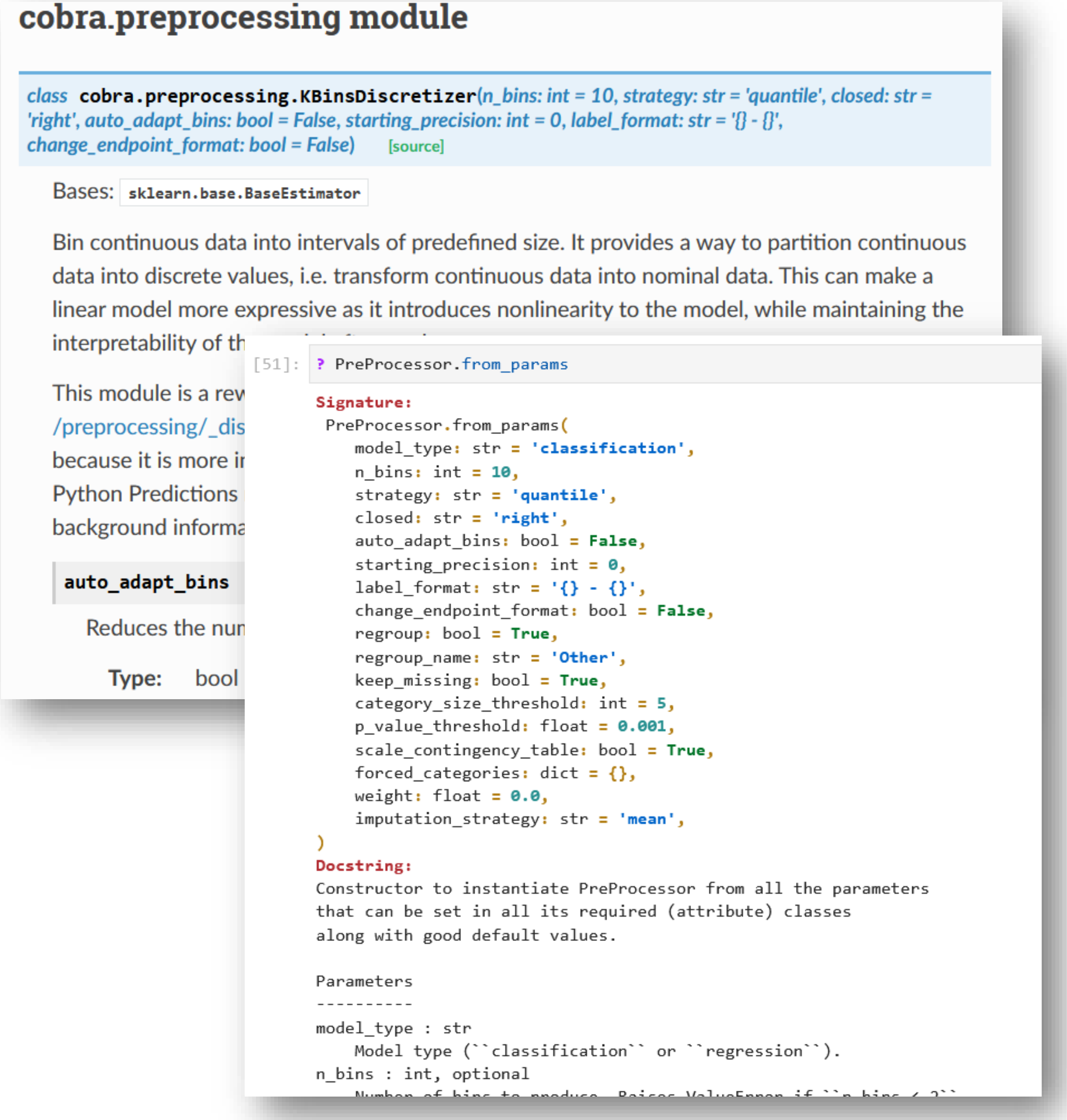
Model building

Evaluation

Industrialization

How can I improve my “standard” Cobra model?

- **Mainly by improving the processing of your input data and feature selection.**
    - The previous slides can guide you (less bins, creating interactions, etc.).
    - Check the warnings Cobra prints in your notebook.
- **Each component in Cobra has a default configuration that you can adapt** – consult the documentation!
    - Online
    - Directly “during coding”: help(module/funtion) or ?module/function
- **Experiment!**
    - Your problem solving course comes in handy!☺
- **A number of advanced features might help you but are not yet integrated in Cobra.**
    - Check the last slides for library suggestions
    - Feel free to commit back to Cobra after your experiments!



cobra.preprocessing module

class **cobra.preprocessing.KBinsDiscretizer**(*n_bins: int = 10, strategy: str = 'quantile', closed: str = 'right', auto_adapt_bins: bool = False, starting_precision: int = 0, label_format: str = '{} - {}', change_endpoint_format: bool = False*) [source]

Bases: `sklearn.base.BaseEstimator`

Bin continuous data into intervals of predefined size. It provides a way to partition continuous data into discrete values, i.e. transform continuous data into nominal data. This can make a linear model more expressive as it introduces nonlinearity to the model, while maintaining the interpretability of th

This module is a rew
/preprocessing/_dis
because it is more ir
Python Predictions
background informa

auto_adapt_bins

Reduces the nur

Type: bool

```
[51]: ? PreProcessor.from_params
Signature:
 PreProcessor.from_params(
    model_type: str = 'classification',
    n_bins: int = 10,
    strategy: str = 'quantile',
    closed: str = 'right',
    auto_adapt_bins: bool = False,
    starting_precision: int = 0,
    label_format: str = '{} - {}',
    change_endpoint_format: bool = False,
    regroup: bool = True,
    regroup_name: str = 'Other',
    keep_missing: bool = True,
    category_size_threshold: int = 5,
    p_value_threshold: float = 0.001,
    scale_contingency_table: bool = True,
    forced_categories: dict = {},
    weight: float = 0.0,
    imputation_strategy: str = 'mean',
)
Docstring:
Constructor to instantiate PreProcessor from all the parameters
that can be set in all its required (attribute) classes
along with good default values.

Parameters
----------
model_type : str
    Model type (``classification`` or ``regression``).
n_bins : int, optional
    Number of bins to produce. Raises ValueError if ``n_bins < 2``
```

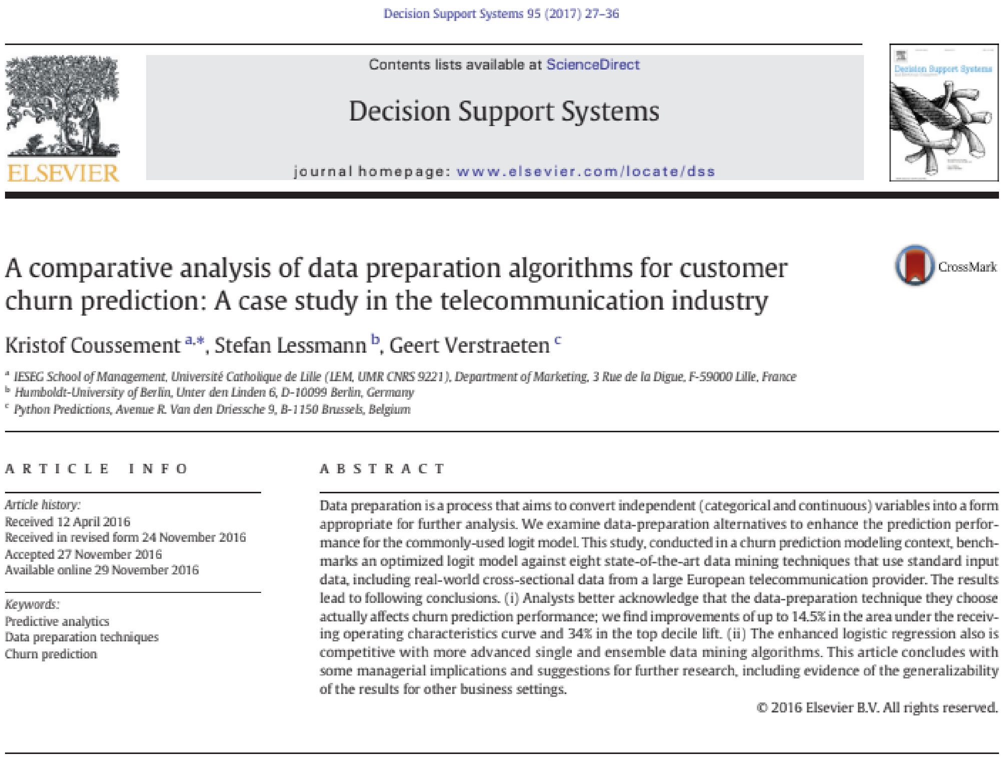
Let’s get our hands dirty!

A tutorial.

Link: GitHub repo

More info

- Github page: https://github.com/PythonPredictions/cobra(contains tutorials too).
- Documentation site.
- Predictive analytics courses (Datacamp, 2 courses)
- Research article on the applied techniques and their performance versus alternatives.



Decision Support Systems 95 (2017) 27–36

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss

ELSEVIER

A comparative analysis of data preparation algorithms for customer churn prediction: A case study in the telecommunication industry

Kristof Coussement [a,*], Stefan Lessmann [b], Geert Verstraeten [c]

[a] IESEG School of Management, Université Catholique de Lille (LEM, UMR CNRS 9221), Department of Marketing, 3 Rue de la Digue, F-59000 Lille, France
[b] Humboldt-University of Berlin, Unter den Linden 6, D-10099 Berlin, Germany
[c] Python Predictions, Avenue R. Van den Driessche 9, B-1150 Brussels, Belgium

ARTICLE INFO

Article history:
Received 12 April 2016
Received in revised form 24 November 2016
Accepted 27 November 2016
Available online 29 November 2016

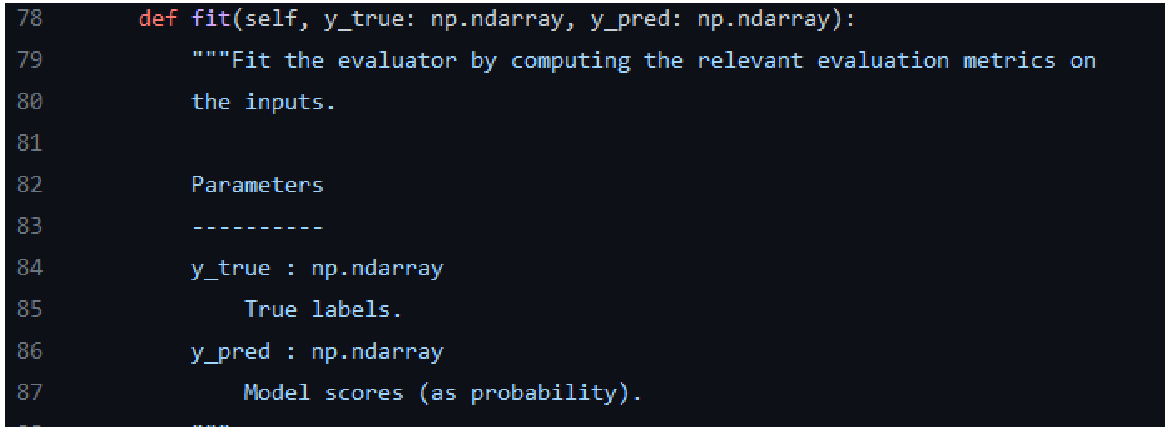
Keywords:
Predictive analytics
Data preparation techniques
Churn prediction

ABSTRACT

Data preparation is a process that aims to convert independent (categorical and continuous) variables into a form appropriate for further analysis. We examine data-preparation alternatives to enhance the prediction performance for the commonly-used logit model. This study, conducted in a churn prediction modeling context, benchmarks an optimized logit model against eight state-of-the-art data mining techniques that use standard input data, including real-world cross-sectional data from a large European telecommunication provider. The results lead to following conclusions. (i) Analysts better acknowledge that the data-preparation technique they choose actually affects churn prediction performance; we find improvements of up to 14.5% in the area under the receiving operating characteristics curve and 34% in the top decile lift. (ii) The enhanced logistic regression also is competitive with more advanced single and ensemble data mining algorithms. This article concludes with some managerial implications and suggestions for further research, including evidence of the generalizability of the results for other business settings.

“This shouldn’t happen…”/ “I’d like…”

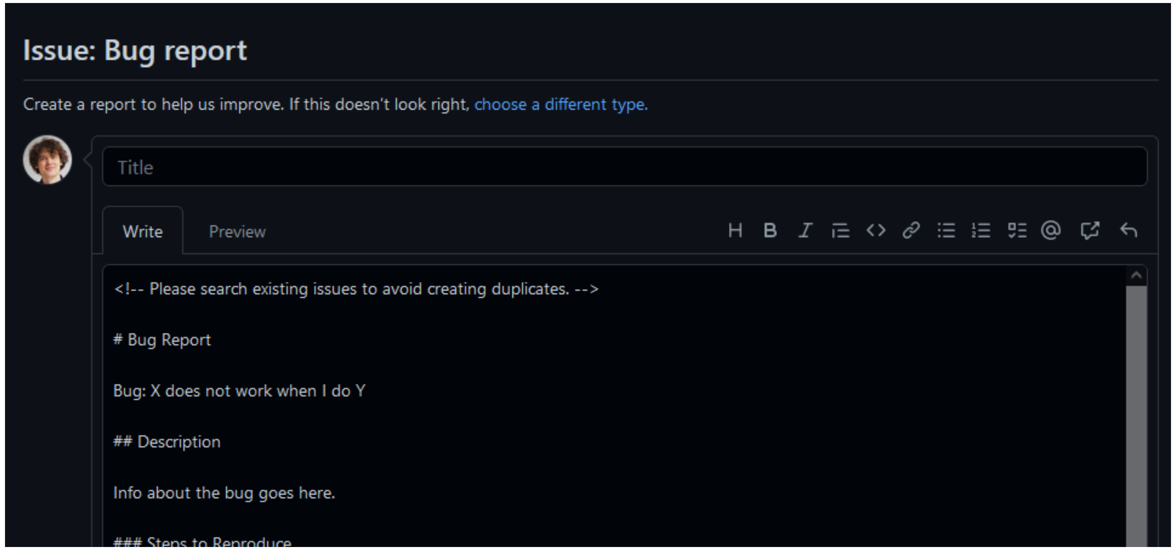
- **If in doubt, you can always also consult Cobra’s source code.**
  - In the Github repository.
  - On your local machine (easy code navigation):
    - Pull the Github repository
    - Browse the code in an IDE such as PyCharm.



```
    def fit(self, y_true: np.ndarray, y_pred: np.ndarray):
        """Fit the evaluator by computing the relevant evaluation metrics on
        the inputs.

        Parameters
        ----------
        y_true : np.ndarray
            True labels.
        y_pred : np.ndarray
            Model scores (as probability).
```

- **You can write Github Issues: https://github.com/PythonPredictions/cobra/issues**
  - Either a **bug report** or a **task**
  - Describe your issue well, provide steps to reproduce!



Issue: Bug report

Create a report to help us improve. If this doesn’t look right, choose a different type.

```
<!-- Please search existing issues to avoid creating duplicates. -->

# Bug Report

Bug: X does not work when I do Y

## Description

Info about the bug goes here.

### Steps to Reproduce
```

Join us if you are interested!

The Cobra wishlist

- **User friendliness:** better support for categorical variables, documentation update, customizing variables on plots, custom #bins per variable, target encoding imputation improvement.
- **More automated tuning** of model hyperparameters
- **Model monitoring:**
- Keeping track of model scores & metadata
    - Allow automated re-fit: pick the best number of features heuristically from the forward feature selection table, rather than requiring manual investigation of the plot.
- **PySpark implementation** (started)
- **No longer require manual tweaking of thresholds in univariate selection:** drop variables if the perform +- equally well to a variable containing random values. (use https://github.com/predict-idlab/powershap)
- **Industrialization:** (started)
    - Cobra deployment as Docker image/Compose file
    - integration with model monitoring databases (postgresql/bigquery/...)
    - Airflow pipelines
    - Dbt integration
- **Using data set split namings consistent with what is used more widely in data science: train/validation/test.**
    - Keeping our namings will only confuse more and more data scientists using our framework and looking up articles such as **https://www.statology.org/validation-set-vs-test-set/**.
- **Improving model insights (interpretability):**
    - Feature contribution per prediction (= per sample)
    - Integrate profiling & profit graphs from pyCaret.
- **Automatic finding model splits**
- **Support cross-validation** (just running Cobra multipe times, on different train-test splits).
- **Automatic feature interactions:**
    - Cobra does not generate them now, whereas a decision tree does
    - On simple problems, adding lots of interactions only marginally improves the model performance (source)
    - If you suspect the problem is non-linear, build a decision tree and add the most important non-linear feature combinations (interactions) in the model. => *Would be good to automate this.*
- **Handling of imbalanced datasets**: promote, streamline & document usage of model_weights hyperparameter.
- **Integrate multivariate outlier detection** (from pyod for example)
- **Automatic model benchmarking**
- Better visibility (linking) towards pythonpredictions.com
- **Various minor improvements** (see github issues)

Contributing guidelines

Our contributors secret Google drive (join the brotherhood to access.)

Join us if you are interested!

What you get back when contributing

- Learning to build strong, interpretable predictive models
- Freedom to add your new ideas
- Writing great code (modules & classes rather than a Jupyter notebook or script + unit tests ☺)
- Exploring pandas, scikit-learn, conda, git, github, CI/CD, etc.

Questions?