Assignment 1 - Some Remarks on Data Preprocessing

Data Science, Oct 2022, gwendolin.wilke@fhnw.ch

These remarks refer to Step 2 ("Data Preprocessing") of the step-by-step-guide published on Moodle.
It comments on some of the questions that came up in the coaching session.

Attributes not available for prediction

It is important to notice that not all of the attributes that are present in the historic data set are available for predicting the interest rate of new loan applications. For example, `issue_d` (the month in which the loan was funded) will not be available in new applications, because the decision on funding will still be open. This means that you cannot use such attributes for learning.

The following is a list of attributes that are not available for new loan applications:

- `collection_recovery_fee`
- `installment`
- `issue_d`
- `last_pymnt_amnt`
- `last_pymnt_d`
- `loan_status`
- `next_pymnt_d`
- `out_prncp`
- `out_prncp_inv`
- `pymnt_plan`
- `recoveries`
- `term`
- `total_pymnt`
- `total_pymnt_inv`
- `total_rec_int`
- `total_rec_late_fee`
- `total_rec_prncp`

Part of the grading of both assignments is a "Reality Check" of your final models (see assignment description on Moodle). Here, we emulate the process of applying your final models to a data set that contains new incoming data, i.e., in the case of assignment 1, new loan applications. In this data set, the above attributes will be present, but will not contain any values.

In order to deal with this issue, it is easiest that you include in the R file that you submit for "Reality Check" a data preparation pipeline (cf. "Reality Check" step 3). Besides other data preparation steps, the pipeline removes the above attributes from the data set that is loaded in "Reality Check" step 2.

Remark: Notice that Tableau Prep Builder will be useful for data exploration and for planning your data preparation pipeline. Yet, you cannot integrate Tabelau in the R file for the "Reality Check".

Manual feature selection

In manual feature selection, we make a manual pre-selection of features that we remove from our data set right away. The fact that a feature does not exist in new incoming data is only one of the reasons to remove a feature. Remember that other reasons exist: some of the features may not be informative (such as `member_id`), some may be sparse (such as `open_acc_6m`), and some would require natural language processing (NLP) to be usable as input variable (such as `desc`).

You may find it helpful to categorize each feature as suggested above and compile a list of features for manual removal. Yet, take some extra attention when dealing with sparse attributes (cf. notes on "different semantics of missing data" below).

Credit history

All attributes regarding the credit history of an applicant (such as `delinq_2yrs` or `open_acc_6m`) are available in the applicant's public credit records. Contrary to my initial assumption it is therefore not necessary to distinguish between costomers who have a history with LC and those who don't.[1]

Joint applications

You may have noticed that `application_type = JOINT` applies only for 13 loans in the historic data set. Yet, if present, it may be pivotal for prediction (or not - we don't know yet). You need to make a decision if you want to use this attribute or not - and how to deal with it.

Semantics of missing data

As a consequence of the the small number of joint applications, all attributes that refer to joint applications (such as `dti_joint`) are sparse. Yet, the values of these attributes are not "missing" in the ususal sense:

- Assume `application_type = INDIVIDUAL` for a certain application. Then the value of `dti_joint` is missing. Yet, it is what we expect. It is ok that the value is missing, because there is nothing in the real world (no joint dti) that could correspond to such a value.
- This is in contrast to data values that do have a real world correspondant, but have not been entered in the data base.

Notice that this is a ubiquitous issue. As a more mondane example consider the follwoing: For tax calculations, it makes a difference if a person is married or not.

- If `name_of_spouse` is missing because the person is not married, it is ok that it is missing, because no spouse exists in the real world. (No data quality issue.)
- If the person is married, but didn't provide the information on her spouse, it's not ok. A spouse exists, but the information was not provided. (Data quality issue regarding missing data.)

Notice that, in our data set, this issue not only applies to attributes related to joint applications. In each such case, you need to decide how to handle it. Common solutions are as follows:

- Exclude the attribute with the risk of loosing important information for prediction.
- Exclude the attribute, but additionally define (one or several) business rules that account for the cases where the attribute applies. An example of a set of business rule for joint applications might be:
  - "If `application_type = 'JOINT'` AND `verification_status_joint = 'Source Verified'`, apply interest rate for individual application."
  - "If `application_type = 'JOINT'` AND `verification_status_joint = 'Verified'`, request additional securities OR increase interest rate by 2%."
  - "If `application_type = 'JOINT'` AND `verification_status_joint = 'Not Verified'`, decline application."
- Impute the missing values with the risk of distorting the results. This particularly applies if the number of missing values is high and/or one doesn't have the ressources to employ elaborate imputation techniques.
- Train different models, one that applies when an attribute value is present, another that applies when it is missing. This is only possible when there is enough training data availbale for both options.

[1] Many thanks to James for the private lesson!

Inconsistent encoding of missing values

Notice that in our data set, missing entries are

- sometimes indicated using the string "NA" (e.g. in `dti_joint`), and
- sometimes not indicated at all - the value is simply missing (no entry).
  (e.g. in `verification_status_joint`).

Depending on your data import options, this can carry over to R. Make sure to take care of this issue in data import or data preparation. Otherwise functions like `is.na()` may not work properly.

How R encodes missing values

- For numerical variables, R encodes missing values as `NA`.
- For categorical variables ("factors"), R encodes missing values as `<NA>`.

The reason for the difference is that, for factors, there is a danger to confuse `NA` with the string `"NA"`. To understand this, remember that factor values ("levels") are strings, such as the levels `"female"` and `"male"` of a factor variable `"gender"`. To make sure `NA` is not confused with the string `"NA"`, R adds the angled brackets when dealing with factors: `<NA>`.

Remark:

- Notice also that `NAN` means something entirely different in R. It means "Not A Number".
  `NAN`, e.g., results from invalid operations such as `0/0`.

Data Type Transformations

To plan your data transformation steps, you may find it helpful to categorize each feature by

- data type, i.e., Character, Numeric, Date, ..., and
- scale of measure, i.e., categorical (nominal, ordinal) or numerical (interval, ratio).

The data type of a variable in the imported data set may not reflect it's scale of measurement. If so, you may want to perform a type transformation, e.g., from `character` to `numeric`, or from `factor` (nominal) to `ordered factor` (ordinal).

For categorical variables, it is also useful to check the number of possible values (`levels`). This is particularly relevant for linear or polynomial regression: if the number of levels is high, the dummy variable approach generates a lot of additional dimensions. In this case you may want to either remove the attribute from your list of input variables, or decrease the number of levels manually in preprocessing.

Feature Scaling (Normalization and Standardization)

Feature scaling usually refers to (min-max-) normalization and (mu-sigma- / z-score-) standardization:

- Normalization transforms the data into a range between 0 and 1: $\frac{x-x_{min}}{x_{min}-x_{max}}$.
- Standardization transforms your data such that the resulting distribution has a mean of 0 and a standard deviation of 1: $\frac{x-\bar{x}}{\sigma_x}$.

When to use what?

- In contrast to standardization, normalization transforms to a bounded range. This can be a problem when new incoming data exceeds this range. Another disadvantages is that normalization ends up with smaller standard deviations, which can suppress outliers.

- Other that that, it mainly depends on the application. E.g., in clustering and PCA, standardization is needed, in image processing and neural networks normalization is required.
- Rule of thumb: When in doubt, standardize the data, it shouldn't hurt.

When to apply feature scaling at all?

- Whether you need to apply feature scaling or not mainly depends on the algorithm you use.
- You need feature scaling for
  - **Distance based algorithms.** E.g., k-means clustering, knn, logistic regression, support vector machines (not covered in this module).
  - **Gradient descent based algorithms**. E.g., neural networks, linear regression based on gradient descent (not covered in this module).
  - **Dimensionality reduction algorithms**. E.g., PCA (Principal Component Analysis).
- You do NOT need feature scaling for
  - **Rule based algorithms**. E.g., tree based methods such as CART (Classification and Regression Trees), random forests and gradient boosted decision trees.
  - **Scale invariant algorithms**. E.g., linear regression based on the analytical solution (as covered in this module).
  - **Algorithms relying on variable distributions**. E.g., naive Bayes (not covered in this module).

What about regression?

Rules of thumb:

- Feature scaling is not necessary for linear regression, but necessary for nonlinear regression.
- Standardize, don't normalize.
- When you are interested in prediction, standardize only the input variables, so that you dont need to backtransform the predicted output values.

Reasons for standardizing variables in linear regression:

- **Interpretability of intercept as mean of target:** When you standardize the input variables (but not the taregt variable), the intercept is the mean of the target variable (because the predictor values are in their means at 0).
- **Interpretability of coefficients as variable importance:** When you use standardized input variables, the size of the coefficients can be interpreted as variable importance. I.e., they tell you which variables have most impact on the target variable.[2]

Reasons for standardizing variables in nonlinear regression:

- **Power terms:** Power terms can introduce multicollinearity, which can obscure the statistical significance of your coefficient estimates. Centering the variables by standardizing them addresses this issue.[3]
- **Interaction terms:** An interaction term (product term of different variables, such as xy) of non-standardized variables can also introduce some amount of collinearity (with the exact amount depending on various factors).

[2] Notice that this is usually not true for non-starndardized inputs, because their scales can be different, and the scales directly impact the coefficients (slopes). Example: Assume you are using the population size of a country to predict its economic power. In that case, the regression coefficient (slope in this direction) is on a very small order of magnitude (e.g., 10^{-9}), because one unit change in population size (1 additional person) increases the economic power insignificantly. This might be very different for another input variable, such as time measured in years. Here a unit change (plus one year) may make a huge difference in economic power (depending on the current economic climate of course…)

Notice also that standarding the inputs decreases interpretability of the coefficients for descriptive analytics: If you want to quantify how much the target variable will change with one unit change of a predictor (e.g., «If I increase adverstisement by 1 million, sales will double.»), you have to rely on non-standardized predictors. Or, as an alternative, you can center the variables (instead of standardizing them) by just subtracting the mean: $x - \bar{x}$.

[3] Example: In nonlinear regression we interpret $Y \sim x + x^2$ as generic quadratic dependency of y on x. But we can alterntivley interpet it as multiple linear regression with two predictors, x and x^2. In multiple linear regression we need to avoid linearly correlated inputs!