Delft University of Technology

Data Science for Finance
WI3435TU

Assignment: Part I. Data-cleaning

Authors:
Dinu Gafton
Dan Sochirca

January 29, 2023

Features with a lot of missing values.

Features with a lot of missing values can reduce the sample size of the data, making it difficult to fit a reliable model. They can also introduce noise and bias into the data. In our dataset, **39 features** happen to have more than 25% missing values, therefore, they are dropped. After this step, the dataset contains *85 features* and *217452 rows*.

Question 1. Filtering on loan status

	Loan Status	Count	Meaning
0	Fully Paid	112409	Loan has been fully paid off.
1	Current	67079	Loan is up to date on current payments.
2	Charged Off	34358	Loan for which there is no longer a reasonable expectation of further payments.
3	Late (31-120 days)	2252	Loan hasn't been paid in 31 to 120 days (late on the current payment).
4	In Grace Period	885	The loan is past due but still in the grace period of 15 days.
5	Late (16-30 days)	461	Loan hasn't been paid in 16 to 30 days (late on the current payment).
6	Default	8	Loan is defaulted on and no payment has been made for more than 121 days.

Figure 1: Different values of loan status and their counts.

After inspecting the value counts of the loan status feature, it was concluded that we should *discard instances* with values: **Current, Late (31-120 days), In Grace Period, Late (16-30 days)**, as they are loan cases in limbo (can be observed in Figure 1).
Remaining instance values of loan status were mapped as following: **'Fully Paid'** as **1**, **'Charged Off'** and **'Default'** as **0**.
After this step, 146775 rows remained in the dataset, so *70677 instances were discarded.*

Question 2. Preventing leakage

In this part of the assignment, the features which represent data leakages have to be discarded because they can lead to models that may perform well on the training data but poorly on new, unseen data.
Since the whole dataset consists of loan applications only from the year 2016, it was easy to find the "leakages" in our data: **out_prncp** - represents the remaining amount of the loan to be paid in the future, **last_pymnt_d**, **last_credit_pull_d** (both of those features have Date values after the year 2016) and **last_pymnt_amnt**.

It can be easily seen in Figure 2 that these features do not provide any meaningful information for our predictive model, thus we have *81 remaining features.*

	name	dtypes	first value	description
33	out_prncp	float64	0.0	Remaining outstanding principal for total …
34	last_pymnt_d	object	May-2018	Last month payment was received
35	last_pymnt_amnt	float64	4361.82	Last total payment amount received
37	last_credit_p…	object	Sep-2018	The most recent month LC pulled credit for…

Figure 2: The Leakage features with their descriptions and first values

Question 3. Dropping features of no or little predictive value.

	name	dtypes	first value	description
0	id	int64	92807938	A unique LC assigned ID for the loan listing.
15	url	object	https://lendingclub....	URL for the LC page with listing data.
32	initial_list_status	object	w	The initial listing status of the loan. Possible values are - W, F
116	disbursement_method	object	Cash	NaN

Figure 3: Features with no/little predictive value

Sometimes, features can have little predictive value and they should be removed because they increase the time and resources required to train the model and could make it prone to overfitting. During inspection of our dataset features, 4 of them got our attention: **id, url, initial_list_status and disbursement_method**. (see figure above) For **id** and **url** it is obvious, they can't be of any help. The **disbursement method** doesn't really say anything about whether the loan will be repaid or not. Finally, the **initial listing status**, which denotes the market in which the loan was listed (W- whole or F-fractional) doesn't imply anything about the borrower, therefore we figured it was best to discard it.
Doing so removed 4 features, therefore *we are left with 77.*

Question 4. Unbalanced features

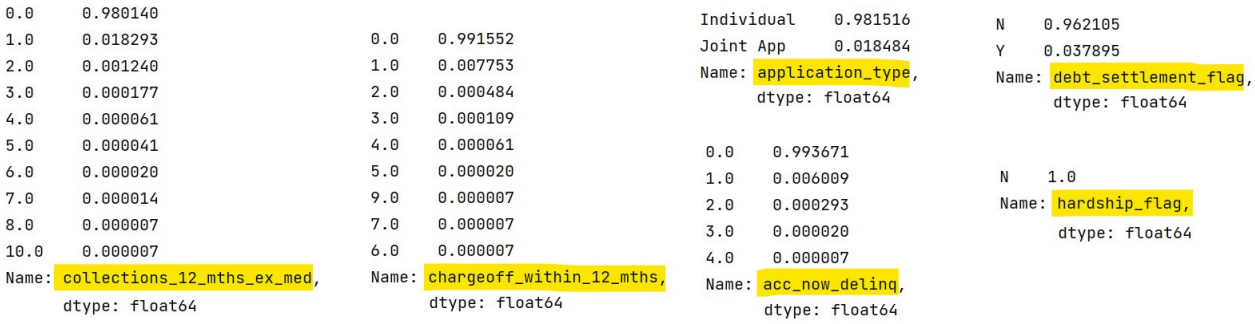

Figure 4: Unbalanced features (and their value percentages) where more than 95% of the instances have the same value

Unbalanced features may cause our model to be biased towards the more frequently occurring class, leading to poor performance on the minority class. Generally, a feature is considered **highly unbalanced** when the majority class represents more than **95%** of the data, so we decided to discard all features that have more than **95%** instances with the same value.
By looking at value counts, **we found 6** of them: *application type, hardship flag, debt settlement flag, acc_now_delinq, collections_12_mths_ex_med, chargeoff within 12 mths.* By looking at their possible values in the figure above we can conclude it is reasonable to get rid of them.
Doing so left us with *71 features.*

Question 5. Highly correlated features

Highly correlated features can cause unstable and inconsistent coefficients in the model, making it difficult to interpret and understand. They can also reduce the generalization ability of the model and increase the complexity and computational expense of training.
In this part, we discard of numeric features, because the correlation coefficient cannot be computed for the categorical features. We decided that any correlation **higher than 0.75** should be considered as a "strong" correlation. Strong correlated features can provide the same information to our model, therefore we decided to remove one of the features from each strong correlation we found in the Correlation Matrix (see Figure 5). For the majority of the features, the reasoning was to remove the feature that is "stronger" correlated with the other features (when choosing which feature to remove).

After analysing the whole matrix, we decided to remove the following features: **installment**, **fico_range_high**, **open_acc**, **revol_util**, **total_acc**, **tot_cur_bal**, **open_il_24m**, **open_rv_24m**, **bc_util**, **total_bc_limit**, **num_actv_rev_tl**, **num_bc_sats**, **num_bc_tl**, **num_rev_tl_bal_gt_0**.

The reasoning for the last 4 mentioned features was different. As we can see in Figure 5, all of those features, together with other 4, are *highly correlated among each other* (this can be seen from the big square with many high correlation coefficients, symmetric on the diagonal). In this case, we decided to drop half of the 8 features, resulting in choosing the above-mentioned features.

Question 6. Getting rid of the remaining missing values

	Number of null values
il_util	19132
emp_title	9586
emp_length	9486
title	8241
mths_since_rcnt_il	3760
mo_sin_old_il_acct	3730
bc_util	1703
percent_bc_gt_75	1644
bc_open_to_buy	1636
mths_since_recent_bc	1539

After finding the columns with missing values (can be seen in the Figure above), we decided to do a mix of dropping features and removing instances. Thus:

We have dropped **il_util** (more than 10% of all the instances have this missing value and it was hard to determine how to replace those missing values) and **title** (this feature is a String object with values and information similar to the **purpose** feature, which has only 12 pre-defined values, making easier to recode and work with).

For the remaining features in the figure, we decided to remove just the instances with the missing values. Since some of them are related to each other, the number of deleted instances was smaller than the cumulative number of instances with missing values (a big number of the deleted instances had multiple missing values). In the end we are left with **55 features and 132460 instances with complete data**.

Question 7. Dates

By looking through the remaining object-type features, only the following were found to represent dates:

	name	dtypes	first value	description
13	issue_d	object	Dec-2016	The month which the loan was funded
23	earliest_cr_line	object	Dec-2004	The month the borrower's earliest reported credit line was opened

By inspecting the **issue_d** feature, we can observe that *its values in the dataset span the year 2016.* (see figure below) This is not useful for our prediction problem, as these values are particular only to 2016 loan applications. Therefore, **we will be discarding the feature**.

Regarding the **earliest_cr_line** feature, it has 641 unique dates, spanning from year 1984 to 2010. This can be useful to prediction. Therefore, **we shall recode it into a numerical feature**.
We did so using pandas, but it would be easier to understand by seeing a code snipped:

```
pd.to_datetime(df['earliest_cr_line'], format='%m%Y', errors='ignore').astype('datetime64').astype(np.int64).astype(float)
```

This is obviously not the best solution, but it works for now. We shall change it in the future if we choose to do so.

The Final Dataset

Before preprocessing, we had a dataset consisting of **124 features** and **217452 instances**.
After cleaning the data, we now have **54 features** and **132460 instances**, sufficient for training and testing our model.

Appendix

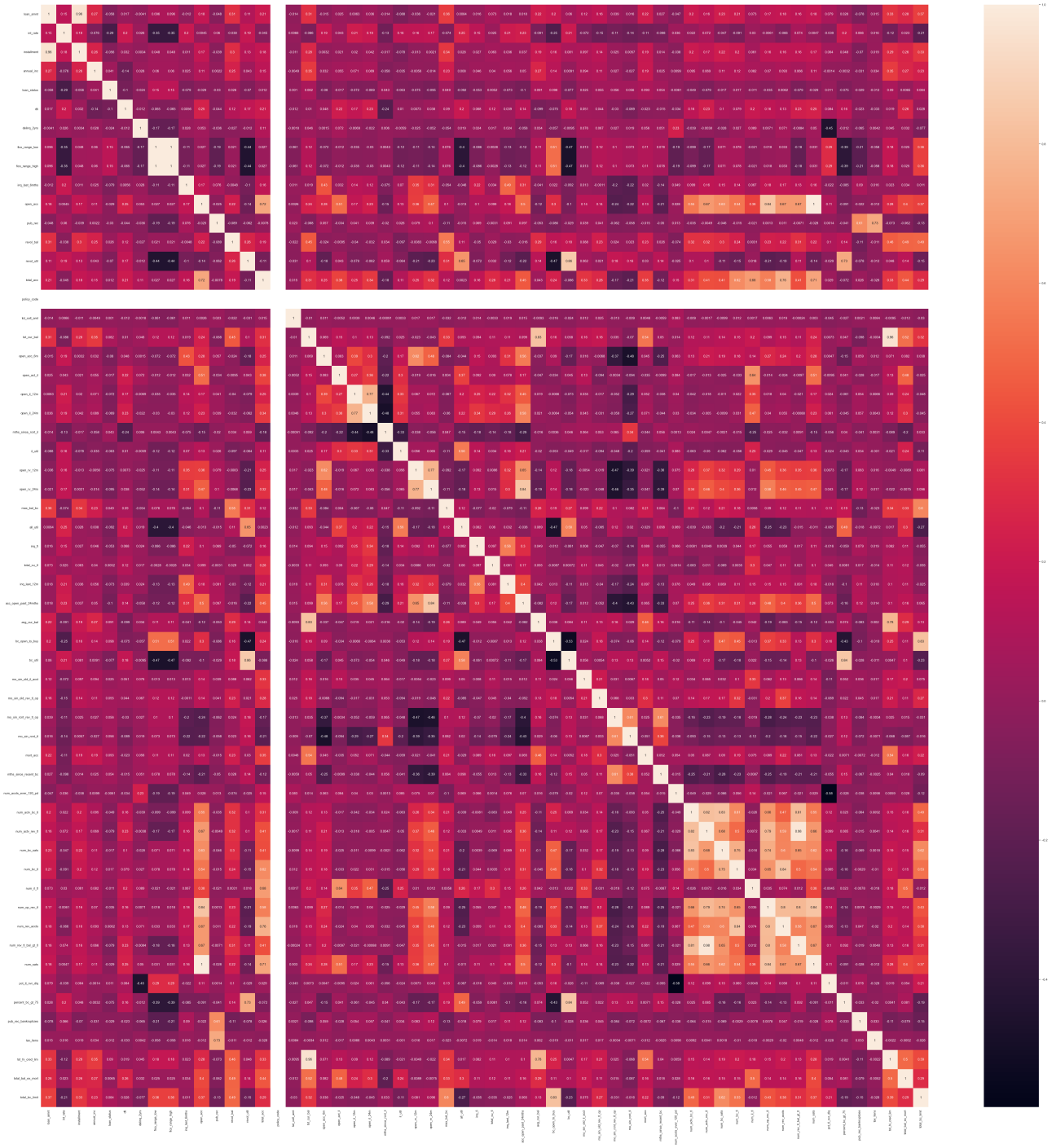

Figure 5: Correlation Matrix of all remaining numeric features