Regression

Denis Tsvetkov and Lukas Ostrovskis

January 2023

1 Splitting the dataset

We have two options in this part:

- making a split and sticking to it for the whole course project
- on each occasion making a split of the data set that seems most suitable

Sticking to a particular split of the data would result in making fair comparisons between the models we will be training - ensuring that data is not a factor influencing the accuracy of the different models would allow us to focus on the differences in the models rather than the data used to train them. However, this would also require us to make a good initial guess about the best overall suitability of the split we choose to stick to from the very beginning, which might have a negative effect on the performance of the models.

On the other hand, making a split of the data set that seems most suitable for each individual model can result in better performance and improve our achieved accuracy. It would, however, make it difficult to fairly compare the performance of different models and might result in us picking a model that is not necessarily more capable but showed better performance due to a favorable data split.

All in all, the two options encapsulate a trade-off between comparison fairness and model performance. Since our main interest in this course project is accuracy and different models may come with different requirements for the data, a one-size-fits-all approach may be too optimistic. Hence, we will try making splits of data that seem most suitable for each model.

2 Recode the categorical variables

After the pre-processing and cleaning of the data, we were left with the following categorical features:

- **term** - nominal variable with only 2 possible values, namely "36 months" and "60 months". For this feature we decided that a simple mapping to the number of months (either 36 or 60) would be better suited than any other numerical transformation;
- **emp_length** - nominal variable with the following values: "<1 year", "1 year", "2 years", ..., "9 years", "10+ years". Here, we again opted for something different than one-encoding: we again mapped to the first number (the years of employment) instead, where "<1" became 0 and "10+" - 10;
- **initial_list_status** - nominal variable with only 2 values - "w" and "f". Again no need for one-hot encoding (as we only have 2 values). Instead we decided to map "w" to 1 and "f" to 0;
- **home_ownership** - nominal variable with 4 possible values - "MORTGAGE", "RENT", "OWN", and "ANY". Here, we decided to introduce 4 new features using one-hot encoding and we also dropped the original **home_ownership**;

- **verification_status** - nominal variable with 3 possible values - "Source Verified", "Verified", and "Not verified". We used one-hot encoding in the same way as for the previous feature;
- **purpose** - nominal variable with 11 different values. We used one-hot encoding in the same way as for the previous features;
- **addr_state** - nominal variable with 49 different values. We used one-hot encoding in the same way as for the previous features;
- **sub_grade** - the only categorical variable that we identified as ordinal with values starting from "A1" to "A5", and similarly for all other letters B-G (inclusive). We decided to map each of those values with the numbers from 1 to 35, where "A1" became 1, "A2" - 2, ..., "B1" - 6, ..., and "G5" - 35.

3 Scale the data

Min-max scaling preserves the original shape of the data distribution, which could be important for some of the models we will be training. However, it is sensitive to outliers, because a single extreme value can drastically change the min-max values of the feature and can therefore make the data set more sparse and cause models to perform poorly.

On the other hand, Z-score scaling is not as sensitive to extreme values, since it only uses the mean and standard deviation to scale the data (see Hull, section 2.11). Though, we have to keep in mind that it does not preserve the original shape of the data distribution, which could result in worse performance of some models.

In our case, a significant amount of the features in the data set contain extreme values. A few of such features can be found in the table below.

Feature	Mean	50th percentile	75th percentile	Max
dti	18.76	18.08	24.81	999
avg_cur_bal	13457.65	7434	18490	463698
pub_rec	0.265	0	0	47
total_acc	24.91	23	32	176
delinq_2yrs	0.35	0	0	21

Table 1: Features with extreme values

Since there are obvious outliers and the value range of the features is not strictly bounded, we decided that Z-score scaling is the better choice.

4 Build and fit the regression models

We built 3 logistic regression models and applied Ridge, Lasso, and Elastic Net regularization techniques to them. By using cross-validation and relying on the misclassification rates, we identified the optimal hyperparameter values for each model: 0.0001, 109.854, and 0.000135 (for Elastic Net we also optimized the l1 ratio and we ended up with 0.9 as the optimal value), respectively. To elaborate more on that, for classification we used a threshold of 0.5 (probabilities higher than that we mapped to 1 (paid out loans), and those below - to 0), and for each model, we relied on approximately 20 hyperparameter values (except for Ridge which had a larger range of values so we decided to use 50) to choose from for the 5-fold cross-validation. Finally, the misclassification rates against the validation sets for the Lasso and Ridge models can be found in the 2 plots below.

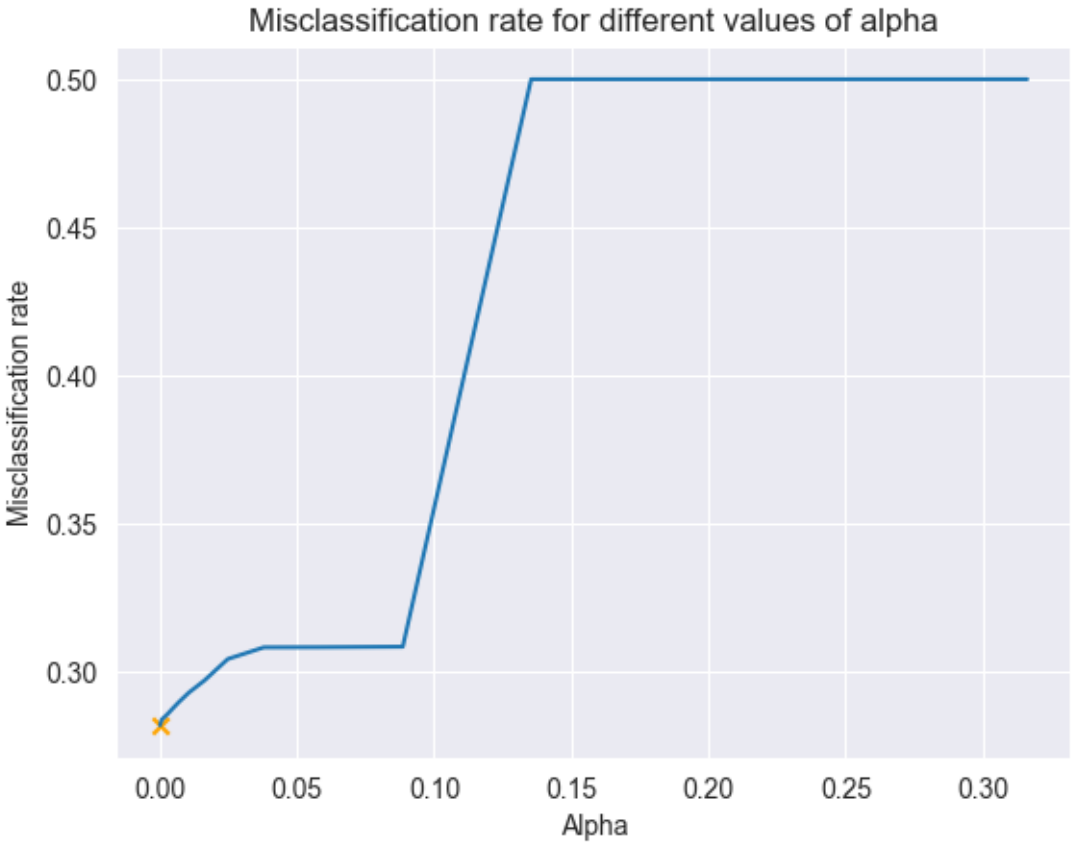

Figure 1: Misclassification rates for different values of the hyperparameter alpha for the Lasso model. Orange cross is the optimal value that was picked.

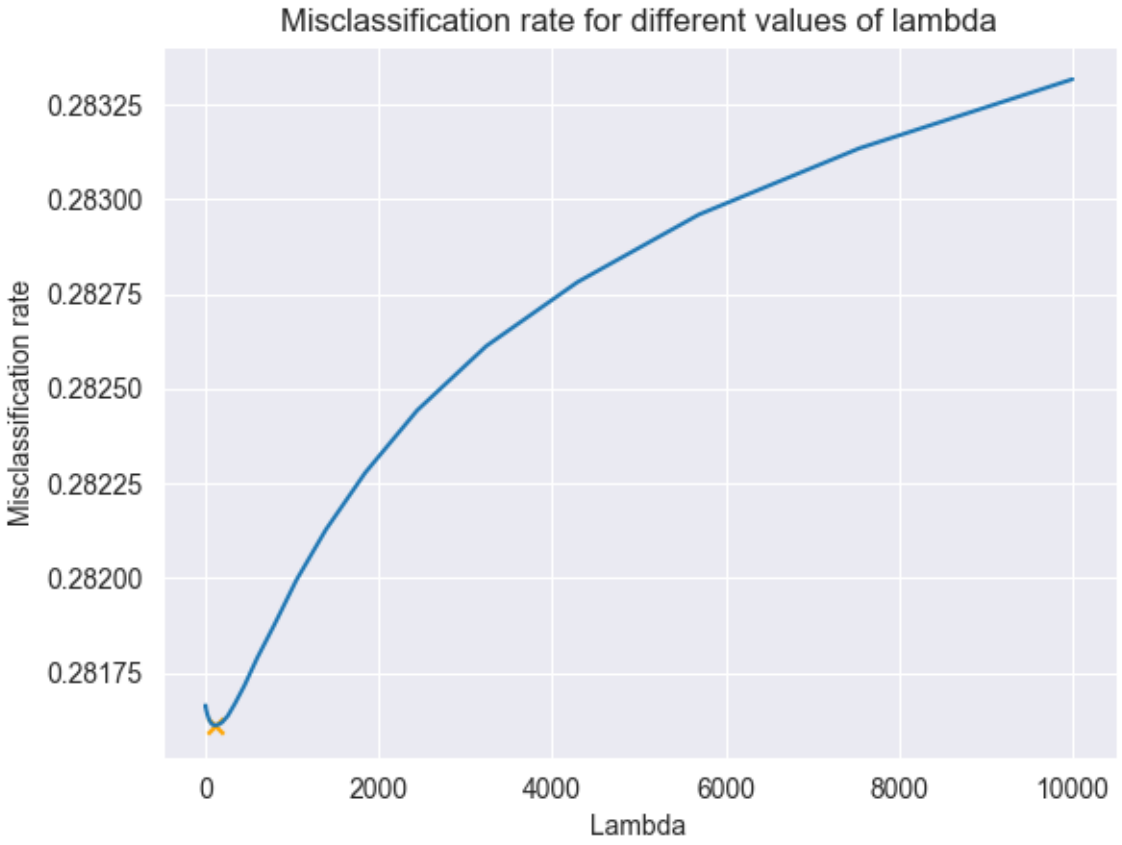

Figure 2: Misclassification rates for different values of the hyperparameter alpha for the Ridge model. Orange cross is the optimal value that was picked.

It is also interesting to analyze which features are deemed "most important" by each of the regularized models. This can be determined by finding the features with the largest absolute coefficient values. The surprise here was that all 3 models had 5 address states in the top 10 most important features, and 10 in the top 20. We decided, to ignore those as they did not seem that valuable to us, and furthermore, they had the same frequency in terms of loan status as the whole data. The features that we ended up with can be seen in Table 2 below.

	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5
Lasso	int_rate	small_business	num_actv_rev_tl	RENT	dti
Ridge	int_rate	small_business	num_rev_accts	term	MORTGAGE
Elastic Net	int_rate	small_business	term	num_actv_rev_tl	RENT

Table 2: Most important features for each of the models

As the three models use different regularization techniques, namely l1, l2, and a mix of the two, it is normal for the models to identify distinct features as important, although they were all trained on the same data. What is important to note is that **int_rate** and **small_business** were the 2 most important features for all of the models. After some analysis on them, we found out that both are of big interest for detecting loans that might default as the frequency of paid loans for **small_business** was significantly lower than that of the whole data (roughly 64% compared to 77%). As for **int_rate**, we noticed that the higher the interest rate was the higher the chance for default became. For comparison: for the lowest interest rate in the data (5.32), the defaulted loans

accounted for 4% of all loans, whereas for the largest interest rate (13.99) - that number increased to 25%.

5 Model comparison

We decided to test 9 different thresholds on our models - 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9. Tables with the confusion matrices for each of the models can be found below.

Note: The confusion matrices generated by the scikit-learn library[1] have a different structure compared to the one mentioned in the book (see Table 3).

TN	FP
FN	TP

Table 3: The format of the confusion matrices' generated by the scikit-learn library

Model / Threshold	0.1		0.2		0.3		0.4		0.5	
Lasso	0	23.23	0	23.23	0.04	23.20	0.49	22.73	2.31	20.93
	0.01	76.76	0.01	76.76	0.01	76.75	0.26	76.52	1.72	75.04
Ridge	0	23.23	0	23.23	0.06	23.18	0.51	22.72	2.29	20.93
	0.01	76.76	0.01	76.76	0.01	76.75	0.25	76.52	1.74	75.04
Elastic Net	0	23.23	0	23.23	0.05	23.19	0.51	22.73	2.32	20.92
	0.01	76.76	0.01	76.76	0.01	76.75	0.24	76.52	1.72	75.04

Table 4: Confusion matrices for Lasso, Ridge, and Elastic Net for the thresholds 0.1, 0.2, 0.3, 0.4, and 0.5

Model / Threshold	0.6		0.7		0.8		0.9	
Lasso	5.95	17.29	11.93	11.31	18.89	4.35	22.44	0.79
	6.07	70.69	17.04	59.72	39.14	37.62	61.36	15.41
Ridge	6.01	17.23	12.03	11.2	18.92	4.32	22.42	0.81
	6.13	70.63	17.04	59.73	39.22	37.54	61.17	15.6
Elastic Net	5.98	17.26	12.02	11.22	18.91	4.33	22.44	0.8
	6.11	70.65	17.1	59.66	39.14	37.62	61.28	15.48

Table 5: Confusion matrices for Lasso, Ridge, and Elastic Net for the thresholds 0.6, 0.7, 0.8, and 0.9

It may appear that a threshold of 0.5 is best for all of the models as it clearly results in the highest accuracy (over 77%). However, it is important to note that this threshold also produces a significant number of false positives, where the model predicts no default but the loan actually defaults.

It is clear that false positives are costly for loan providers. We can also support this claim by referring to Hull (see Hull, section 3.11), who states that revenue can be maximized by picking a threshold such that $1000 \cdot TP - 4000 \cdot FP$ is maximized. Using the formula, we calculated the optimal threshold for all of our models to be 0.8.

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html`

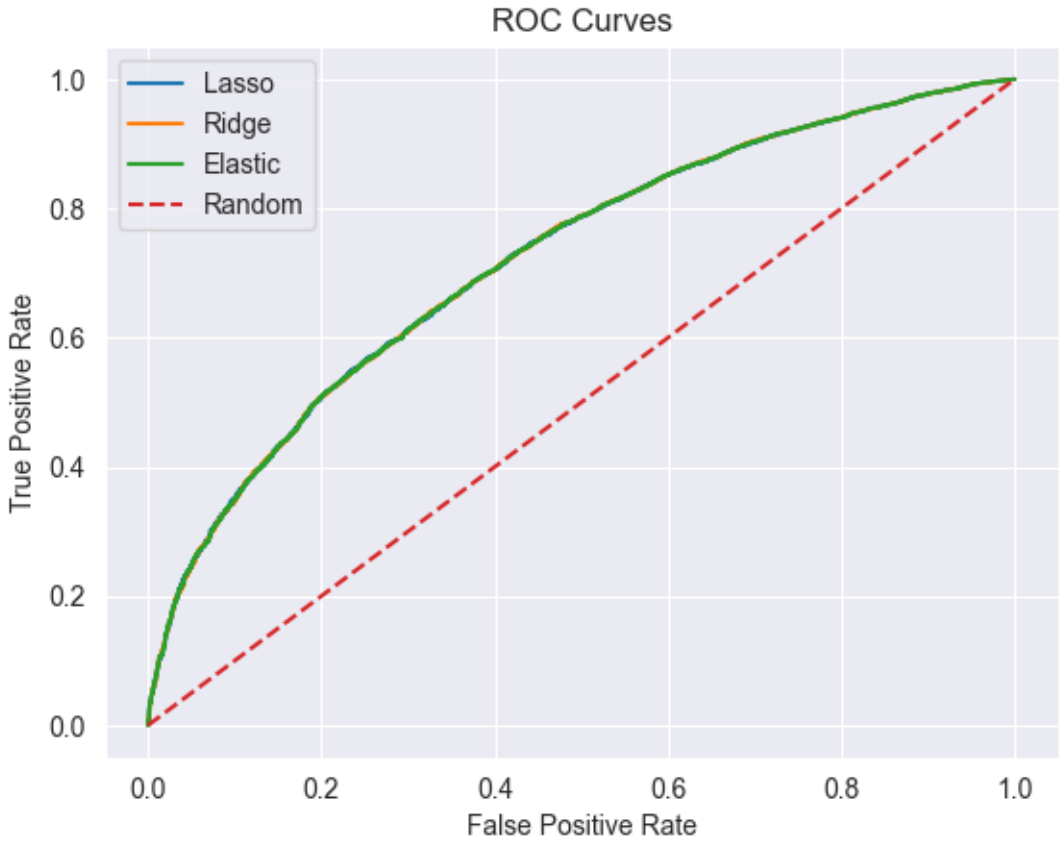

Figure 3: ROC for Ridge, Lasso and Elastic Net models

For a more accurate comparison, we can consider the areas under the curve (AOC) of each of the models. They are:

- Lasso - 0.71966;
- Ridge - 0.71976;
- Elastic Net - 0.71973.

Hence, based on the AUCs we can conclude no model is better than the other two and that they are equally viable options for predicting loan statuses.

References

[1] Hull, J. C. (2019). Machine Learning in Business: An Introduction to the World of Data Science.