Learning from small datasets

PV115 Laboratoř dobývání znalostí

Ján Krčmář, 506450

Dataset and task description

The “Don’t overfit II!” is a Kaggle competition with a **binary classification task** on a rather unusual dataset. It contains **300 training samples** and **19700 testing samples**, each with **299 continuous attributes**, from a normal distribution.

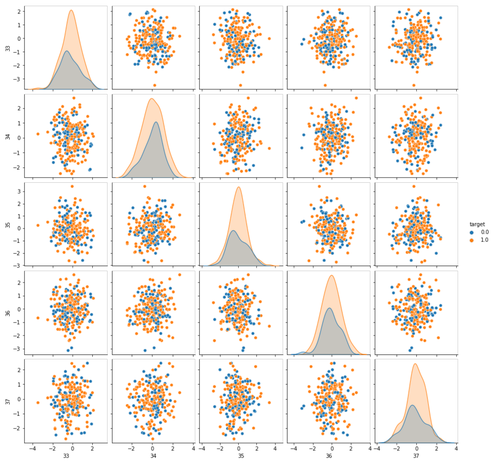

Fig. 1.: Pairplot with randomly selected attributes.

Based on these properties is it of the highest necessity to come up with a solution that does not overfit, but manages to learn the pattern in limited data. In this experiment, the evaluation metric of choice is the Area Under the ROC Curve (ROC AUC).

Base models

For the small number of training samples, we tried out multiple models with fewer parameters that can be regularized to minimize the chance of overfitting. We also experimented with Random Forest (as an out-of-the-box ensemble method) but we decided against it afterwards.

	Public ROC AUC	Private ROC AUC
LogisticRegression	**0.6521**	**0.6685**
RandomForestClassifier	0.5167	0.5223
GradientBoostingClassifier	0.6114	0.6001
DecisionTreeRegressor	0.5102	0.5133
NaiveGaussianBayes	0.5973	0.5934

Table 1.: Sklearn classifiers with default settings evaluated on the public and private subsets of test dataset.

Preprocessing Pipeline

When training on such a small dataset, the influence of each individual example on the model is notable. Picking the right ones is therefore a crucial step in the preprocessing stage. Yet we had to be very careful not to remove too many instances in order to keep the training dataset size as large as possible. Each step of the pipeline was fitted on all training examples and subsequently applied on the current sample of data in the training loop.

scaling

Robust Scaler

Robust Scaler centers and scales the data feature-wise according to the quantiles.

↓

class balancing

SMOTE + TOMEK

SMOTE is an over-sampling method generating new samples without acknowledging the underlying distribution. In case of overlapping classes there are many noisy samples generated. **TOMEK** is an under-sampling method that is used to clean noisy samples.

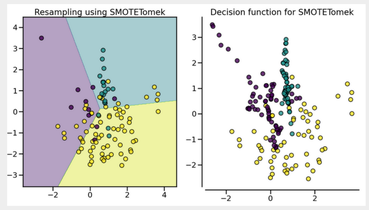

Fig.: Visualisation of SMOTE + TOMEK hybrid method.
Source: imbalanced-learn.org

↓

outlier removal

Isolation Forest

Isolation Forest returns an anomaly score for each sample by splitting randomly selected features until the given sample is isolated. The number of splits required averaged over a forest of isolation trees serves as a measure of normality.

Since our data come from normal distribution there are not many obvious outliers. Using this method to remove a fixed number (1%) of samples improved the overall ROC AUC metric .

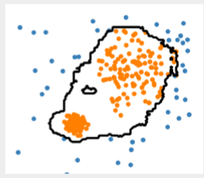

Fig.: Visualisation of Isolation Forest
Source: scikit-learn.org

↓

feature selection

RFECV

Recursively considering a smaller and smaller set of features each time, the **Recursive Feature Elimination** evaluates current set of features using cross-validation.

In our study 12 - 15 features (out of 299) were selected in the best performing models. In standalone experiments feature elimination alone improved the ROC AUC metric (for most linear classifiers) by more than 7%.

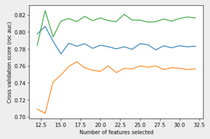

Fig.: Visualisation of RFECV with a 3-fold CV.

Fig. 2.: Preprocessing pipeline.

Training

The training / model finding strategy we used is the **Stratified Shuffle Split** from scikit-learn. In each iteration indexes for train and validation samples are picked in a way that the class ratio is preserved. We split our training data into train and validation subsamples and feed it to a grid search cross-validation where the optimal model hyper-parameters are found. The table below provides an overview of grid search parameters for a Lasso classifier.

	Values
alpha	[0.022, 0.021, 0.02, 0.019, 0.023, 0.024, 0.025, 0.026, 0.027, 0.029, 0.031]
tol	[0.0013, 0.0014, 0.001, 0.0015, 0.0011, 0.0012, 0.0016, 0.0017]

Table 2.: Grid search parameters for Lasso classifier.

Combining Models

Each training loop (on a given fold of data) produces one model with it’s best hyper-parameters. This model is evaluated on the validation samples. (We use 35% of the original training samples for validation in the training procedure.) If the current model has a R2 score of 0.2450 or higher we use it to predict the public and private test sets. In each iteration of training, when a new model passes the validation threshold, it’s predictions are appended to the ones from the previous (successful) model. After the training finishes, the final probability of each test sample is computed as the mean probability of each model’s prediction (each “successful” model predicts the probability of a given example belonging to the positive class).

Results and discussion

	Public ROC AUC	Private ROC AUC
LogisticRegression	0.6521	0.6685
Lasso	0.5000	0.5000
Lasso Ensemble (no preprocessing)	0.7994	0.7890
Preprocessing + Lasso Ensemble	**0.8270**	**0.8320**
LB Probing	**0.8900**	**0.8690**
Small dataset learning	0.859	0.859

Table 3.: Competition results

Our results are summarized in the table above. It is important to take into account that our scores are not directly comparable to the ones from the competition, since our experiments were done on a different dataset (generated using the same ideas).

From the competition, the LB Probing method was the winning submission that used the information from submitting test results to tune coefficients for each of the 100 most important attributes. Rafael Alencar (who ended on the 43. position) used a similar technique to ours, but probably managed to fine tune individual steps better.