EE559 Project: German Credit

Ming-Chang Chiu 6947046287 mingchac@usc.edu

May 2, 2017

1 Abstract

In this work, I designed a pattern recognition system to analyze and predict if a user is a potential good or bad customer based on German Credit dataset from UC Irvine. Learning algorithms includes common classifiers such as Naive Bayes (NB), k-Nearest Neighbors (k-NN), Support Vector Machine (SVM) and tree classifiers like Random Forrest (RF) and also the recently popular Neural Network classifier. For evaluation, 20% of the original dataset is set aside as test dataset at the beginning. And as for model selection and dimensionality reduction (DR), I implemented two different schemes. The first one performs DR before cross-validation (CV) hyper-parameter grid search; the second applys cross-validation for both dimensionality reduction and grid search. The result shows that even though the second scheme is more standard and more rigorous, the first scheme achieves no worse performance in this work.

2 Approach

Before doing anything, identify that our goal is to do classification and so look only into classification algorithms. Also, both ordinal and unordered categorical data is present, late fusion should be the best method, but the time complexity due to training different classifiers in serial would be an issue. At the end, I choose to use simpler pattern recognition flow– firstly, set aside test set; secondly, perform normalization; thirdly, dimensionality reduction and training; and finally, model selection.

3 Preprocessing

Since both numerical and categorical features are in the dataset, we need to clean and reformat the data before getting into more advanced analysis. Numerical data is treated as their original form at this stage. Some categorical features are perceived as ordinal so I assign order to them, for instance, in dataset 1, checking account with "little" money is given 1 and "quite rich" 3; yet some are unordered features, for example, sex, so "male" will be cast into one-hot form [1,0] and female [0,1].

Sometimes among a ordinal feature, different category can have the same order. An example would be in dataset 2, as long as the sample has an "other installment plan", no matter it is "stores" or "bank," I assign 1; otherwise, 0.

At the end of this stage, I will concatenate all of them to one big matrix. Note that there will be additional dimensions due to the feature expansion for unordered categorical features.

4 Features Used

For dataset 1, I choose to include all features in the dataset for comprehensiveness then perform dimensionality reduction to get the final training set fed into classifiers. More details on feature selection/dimensionality reduction will be addressed in the next section.

For dataset 2, I use all features in the same way except for "Purpose" because this feature does not seem pertinent to me and there are 11 categories in this feature. I do not want to have too many dimensions so I eliminate this feature at preprocessing stage.

5 Dimensionality Reduction/Normalization

Some learning algorithms are sensitive to feature scaling, so normalization is very important. Normalization I used is simple–standardize features by removing feature mean and scaling to unit variance.

As for dimensionality reduction, I applied two common dimensionality reduction methods, Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) before feeding training data into classifiers. To be brief, apply PCA or LDA on training set and then get a transformed training set, which is later fed into classifiers.

In this work, I experiment dimensionality reduction differently. The first experiment is to perform dimensionality reduction on all training set without cross-validation before classifiers, all training data get the same reduction; the second is more rigorous, perform cross-validation in both dimensionality reduction and classifier stages, so the data fed into classifiers is transformed based on the result of cross-validation. The first scheme is simpler and faster but less exhaustive and rigorous.

6 Classifiers

All classifiers chosen in this work are very common. Scikit-learn provides very well-organized and easy-to-use API for users to call. I choose 7 classification algorithms.

For distribution-free algorithms, I choose k-NN, Linear SVM (L-SVM), and RBF SVM. For statistical classifier, I choose only Naive Bayes (NB) with likelihood features assumed to be Gaussian.

In addition, I decide to try a simple algorithm from recently popular Neural Network, which is Multilayer Perceptron (MLP). MLP trains using back-propagation and can fit non-linear model.

Last but not least, as said in section 1, late fusion should be the best way to tackle feature space in this work, since I choose not to implement it, I use two ensemble-based classifiers as alternatives, Random Forrest (RF) and AdaBoost. Ensemble methods sometimes use multiple algorithms together to get better predictive performance, so they are very suitable for this work.

In Scikit-learn library they are KNeighborsClassifier(), SVC(kernel="linear"), SVC(kernel="rbf"), RandomForestClassifier(), MLPClassifier(), AdaBoostClassifier(), GaussianNB().

7 Evaluation & Dataset Usage

To get as close to real life situation as possible, I set aside 20% of the whole dataset as test set as soon as getting preprocessed data. For consistency, I set a random seed when sampling the test set so the two scheme mentioned in section 5.

At fine-tuning grid search, I use 4-fold cross-validation and compare their mean cross-validation accuracies, then choose the best parameters for that classifier; Scikit-learn provides good interface for user to do this. To save time for running the whole experiment, I only perform cross-validation once for each parameter setting (but actually run many times in reality to check robustness) Grid search starts from exponential search space then reduce to certain linear space.

The baseline is 69.75% for training and 71% for testing (hidden beforehand). Test set will be transformed by the dimensionality reduction paradigm chosen throughout training stage. And the final performance is measured on the test set with best training parameters.

8 Results

Table 1

Best Model for each Classifier (Dataset 1 scheme 1)					
Classifier	Test Acc (%)	f1 (class 1)	f1 (class 2)	CV Acc (%)	DR Method (dim)
k-NN	71.5	0.81	0.45	75.25	LDA (5)
L-SVM	73.5	0.83	0.44	75.125	LDA (5)
RBF SVM	74	0.83	0.43	73.625	PCA (15)
RF	72.5	0.81	0.52	76.125	LDA (8)
MLP	72.5	0.8	0.54	76	LDA (10)
AdaBoost	**75.5**	**0.84**	**0.51**	**71.5**	**PCA (15)**
NB	73.5	0.82	0.5	74.375	LDA (5)

Table 2

Best Model for each Classifier (Dataset 1 scheme 2)					
Classifier	Test Acc (%)	f1 (class 1)	f1 (class 2)	CV Acc (%)	DR Method (dim)
k-NN	73.5	0.83	0.45	74.125	LDA (5)
L-SVM	73.5	0.83	0.44	73.25	LDA (5)
RBF SVM	73.5	0.82	0.52	74	LDA (5)
RF	73	0.81	0.47	75	LDA (9)
MLP	73	0.82	0.44	74.25	LDA (7)
AdaBoost	73	0.82	0.47	70.625	PCA (8)
NB	73.5	0.82	0.5	73.875	LDA (5)

Table 3

Best Model for each Classifier (Dataset 2 scheme 1)					
Classifier	Test Acc (%)	f1 (class 1)	f1 (class 2)	CV Acc (%)	DR Method (dim)
k-NN	71.5	0.82	0.31	72.75	PCA (20)
L-SVM	73	0.83	0.31	71.375	PCA (18)
RBF SVM	70.5	0.81	0.32	74.5	PCA (13)
RF	71.5	0.8	0.49	74.875	LDA (14)
MLP	71.5	0.8	0.48	74.375	LDA (10)
AdaBoost	71	0.8	0.47	72.5	LDA (5)
NB	70.5	0.81	0.38	73.75	LDA (5)

Table 4

Best Model for each Classifier (Dataset 2 scheme 2)					
Classifier	Test Acc (%)	f1 (class 1)	f1 (class 2)	CV Acc (%)	DR Method (dim)
k-NN	70.5	0.8	0.42	69.875	LDA (5)
L-SVM	73	0.83	0.27	70.375	LDA (5)
RBF SVM	70.5	0.81	0.32	74.375	PCA (13)
RF	**73.5**	**0.83**	**0.38**	**74.875**	**PCA (17)**
MLP	71.5	0.81	0.46	71.875	LDA (24)
AdaBoost	71	0.8	0.47	68.125	LDA (5)
NB	70.5	0.81	0.38	71.25	LDA (5)

Note: Best parameters are not listed due to space limit but can be obtained by running code

Plainly judge from performance. I would choose AdaBoost with parameter settings in Table 1 for dataset 1, and Random Forrest in Table 4 for dataset 2. Note that the f1-score and accuracy would be the same as in table.

9 Interpretation

After experimenting so many different algorithms and parameter combinations, among all classifiers, the best is AdaBoost for dataset 1, RF for dataset 2. For all classifiers, their best accuracies is around low 70's, the accuracy no matter in final testing stage or cross-validation stage remain the same, meaning German Credit is a relative hard dataset, 70% may be a bottleneck. Further improvement is possible by more extensive fine-tuning, especially for **dataset 2 and Neural Network classifier**, but can improves only slightly. One thing should be noted that the test set is sampled randomly but using a random seed, if I remove this constraint, sometimes the classifiers can achieve better performance.

Observe from section 8 DR method, LDA seems to be a more desired DR method. The reason my be since we are doing supervised learning, LDA is known to be suitable with labels provided. Also, most algorithms using LDA to achieve best performance reduce dimensionality to 5, the lower bound I set, meaning there are still space to explore in the future and LDA can serve as a good tool for DR in supervised learning.

A quick recap to my two different DR scheme, scheme one is simpler, doing DR before CV; the other more formal and rigorous. From our intuition, the more rigorous, the better, but judge from section 8, the statistics show that the two schemes are comparable and using AdaBoost in scheme 1 for dataset 1 even achieves the highest accuracy. Based on that, when in reality, for saving time purpose, we can use less rigorous evaluation paradigm to get quick result and still get good estimates.

As for F-measure, we want the score to be as large as possible. For class 1, the f1-score does not change a lot, yet for class 2 it does. When choosing the best classifier, f1-score can be taken into account but in this work I did not.

10 Acknowledgement

In this work, code is written in Python, all of the classification methods come from Scikit-learn library, and CSV handling and array processing are based on Pandas and Numpy. In addition, I started the project with reference to a Scikit-learn article "Sample pipeline for text feature extraction and evaluation," enlightening me how to combine stages of evaluation into a Pipeline and providing me some inspirations on Scikit-learn usage.