Glass Type Classification

Machine Learning Engineer Nanodegree

Definition

Project Overview

In forensic science it is important to identify evidence that collected from crime scenes. Shards of glass are common to be found in crime scenes; if these glass samples collected from the crime scenes can correctly and effectively identified, it can help the criminological investigation. Building windows, container, tableware and many other glass products are usually made of different material or different type of glass. By checking their chemical element composition, we shall able to identify where those the shards of glass originally come from. This will in turn boost the forensic science and criminology investigation.
In this project, I will use a public available data set to build a model to identify the type of the glass sample by their chemical or physical attributes. The data set was originally used by Ian W Evett et al in a paper with similar task. However, that paper was published in 1987, the approaches used in the paper are more likely to be out of date. With development of both computer science and machine learning, this project can implement newly developed techniques to see whether we can out-perform the result of the original paper.

Problem Statement

In summary, the problem is to identify the origin of each glass sample. The dataset provide 9 features such as reflection rate, amount of Mg, Ca, Na etc. Each sample is labeled with its type. We have 7 types of glass:

- 1 building_windows_float_processed
- 2 building_windows_non_float_processed
- 3 vehicle_windows_float_processed
- 4 vehicle_windows_non_float_processed (none in this database)
- 5 containers
- 6 tableware
- 7 headlamps

Unfortunately, dataset does not include any sample of glass type 4. The dataset has 214 observations which kind small. This may cause trouble in building the correct model. I will try several different algorithms to test which machine-learning algorithm will fit the task best. In addition, like the author of the original paper, I also separate the dataset into groups. Since different type of glass can fall into same group, both building windows and vehicle windows are windows, we can thus try to identify windows glass from non-windows glass. The original author also provided these results, which can be used as a benchmark for this project.

Metrics

In multi-classification phase, because of the imbalance of the labels, accuracy will not that meaningful. Recall and precision will be more meaningful to measure the outcome. Specifically, I use f1_score to incorporate both measures in one number to quick measure the performance as well as implement the grid search. In addition, confusion matrix is also used to inspect the performance in detail.

In binary case, I will also use ROC to have visual look on the performance. The results will also be present in a format that are comparable to the benchmark.

Analysis

Date Exploration

First, let us have sneak peak of the data set:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

The first column, variable RI, stand for reflection rate, other variables are amount of each chemical element. Variable Type indicate what type of glass each observation is. Further, we can do some descriptive statistics:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
count	214.0	214.0	214.0	214.0	214.0	214.0	214.0	214.0	214.0	214.0
mean	2.0	13.0	3.0	1.0	73.0	0.0	9.0	0.0	0.0	3.0
std	0.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	2.0
min	2.0	11.0	0.0	0.0	70.0	0.0	5.0	0.0	0.0	1.0
25%	2.0	13.0	2.0	1.0	72.0	0.0	8.0	0.0	0.0	1.0
50%	2.0	13.0	3.0	1.0	73.0	1.0	9.0	0.0	0.0	2.0
75%	2.0	14.0	4.0	2.0	73.0	1.0	9.0	0.0	0.0	3.0
max	2.0	17.0	4.0	4.0	75.0	6.0	16.0	3.0	1.0	7.0

Here are the amount of each label (no observation has label 4):

Type	**1**	**2**	**3**	**5**	**6**	**7**
counts	70	76	17	13	9	29

From the table, we see that type 6 (tableware) only has 9 observation, first two types, (building windows) have together over 100 observations. Clearly, the dataset has very imbalanced labels.

3

Exploratory Visualization

We see that the dataset has no missing value, and may have some scale problem. We can further explore the data by do some visualization.

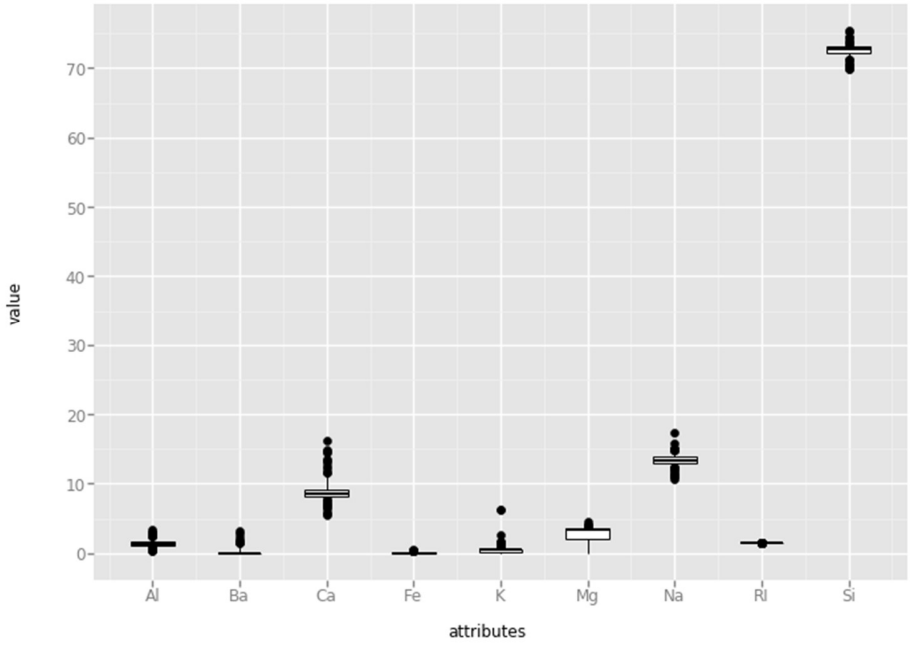

Figure 1 box-plot before scaling

From the graph above, we see the scale issue is significant. After transforming the data to same scale, we have a graph like below (The graph was drawn over training subset):

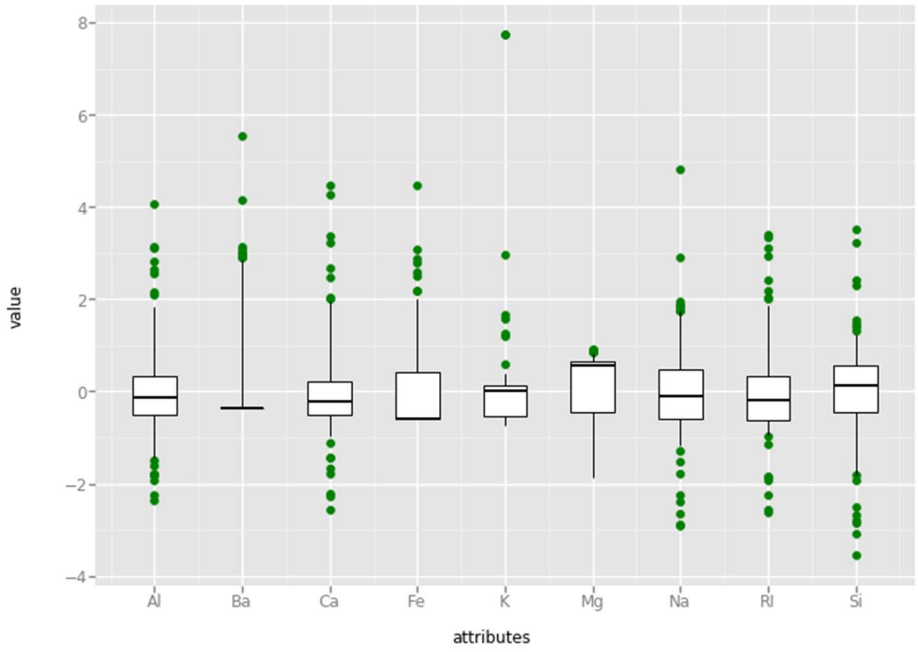

Figure 2 box-plot after scaling

From the box plot above, we see many outliers; however, because of the imbalance labels by removing the outlier, we may just remove the minority class. Also because of the small dataset we have, removing these data, we will have even fewer data to analyze. Therefore, I keep those data intact. We can also have a look on the imbalanced labels visually.

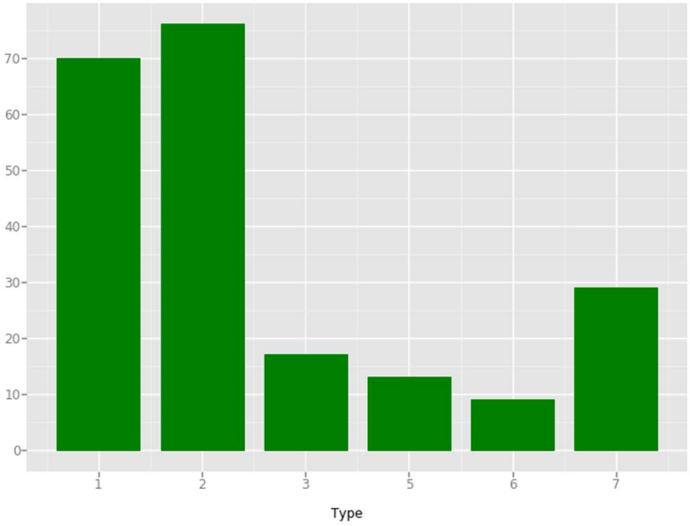

Figure 3 bar-plot for types

We see from the data above the first two class (building windows) comprise the majority of the dataset.

Algorithms and Techniques

In this project, I will using three different supervise classification to classify the glass samples. The algorithms are SVM, KNN, and Random Forests. KNN was used in original paper as well. In addition, the author claimed that KNN was the best the approach to classify the glass sample. SVM and Random Forests are also the well know and more sophisticated classification algorithms. Therefore, we can explore in this project that whether SVM and Random Forests can outperform KNN.

SVM algorithms is the so-called max-margin algorithms. Put it simply, the algorithms try its best to find the "clear cut" boundary, by which to avoid the "near miss" situation. The graph below show the concept visually (graph from Wikipedia.org)

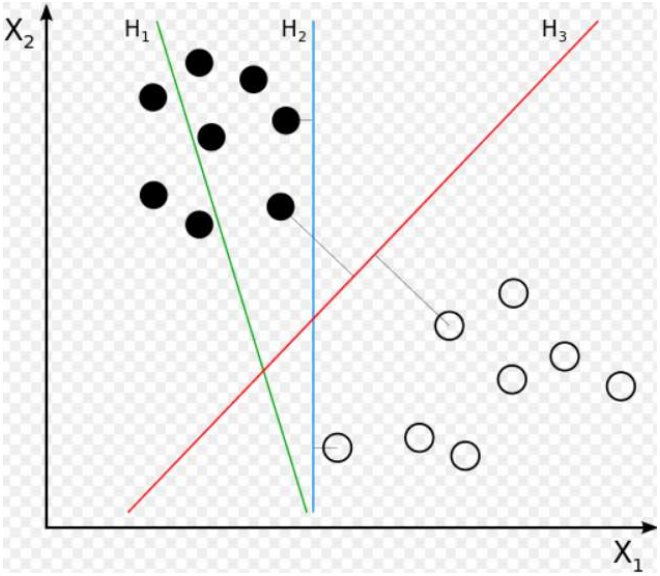

Figure 4 SVM boundary

In the graph, H_2 is the "near miss" situation, and the H_3 is the SVM boundary.

SVM by itself is a linear classifier; however, with help of so-call kernel trick, the SVM can be used in non-linear situations. The kernel transform the data into higher dimension, after which the data become linear classifiable.

Although, SVM was design to tackle binary classification problem, so called one vs rest approach can be used to tackle multi-class classification problem.

Random Forest is the ensemble counterpart of decision tree. The ensemble approach will ameliorate the linear boundary of decision tree algorithms, and have higher accuracy.

A deep decision tree tend to overfit the training dataset, i.e. low bias, but very high variance. Random forests averaging the result of the multiple trees in order to reduce the high variance. The random forests applies bootstrap aggregating to tree learners, by which, the variance will decrease without increasing the bias. In addition, random forests will randomly subset the features at each split. This can avoid that certain dominant features get selected too often, which will cause the correlation between trees and harm the accuracy.

As observed earlier, the data has scale issue; all input data will be preprocessed before implement the algorithm.

Benchmark

The benchmark used in the projects is the result provide in original paper mentioned earlier. However, the author did not use a subset to test the result, rather the entire dataset. Following is the same result provided in that paper, later I will compare the result with the outcome of this project.

Whether are windows:

Number of incorrect answers			
True of Samples	**Beagle**	**Nearest Neighbor**	**discriminant analysis**
Windows(163)	1	5	5
Non-windows(51)	7	7	7

Whether are float processed (only windows are considered):

Number of incorrect answers			
True of Samples	**Beagle**	**Nearest Neighbor**	**discriminant analysis**
Float(87)	10	12	21
Non-Float(76)	19	16	22

For the multi-classification phase, the author just used 2 sample from each category to test the performance of each approach. Since a test on 2 samples can be correct or incorrect just out of chance. I will use a cross-validation with f1_score to see how good each algorithm can be.

Methodology

Data Preprocessing

The data set was separated into training and testing datasets with ratio of 0.8. The data inputs are normalized using sklearn scale function. This will prevent one feature dominate others. There is no categorical attributes, so no encoding is used.

During the process, PCA and LDA are considered, however, no improvement was achieved by implement these dimension reduction techniques. Therefore, these techniques were not used.

Implementation

All algorithms were implemented with scikit-learn. After the data was preprocessed, the X_train, y_train subsets were plugged in the fit method of each algorithm object. Each algorithm object was created by default or some commonly used parameters, and the grid search was used to find best parameter. The score function in the grid search was generated by make_scorer function using f1_score. To adjust the imbalanced labels, average was set to "weighted" in f1_score function.

I repeated the same approach for SVM, Random Forests, and KNN.

After multi-classification phase, I removed the original label, and re-labeled the data for each binary classification task. First, I address the task of identifying windows from non-windows samples. For each observation has label 1, 2, 3, I add label 1, and for all other observation I add label 2 (This due to no observations in dataset has label 4). For this new data set, I repeat what did earlier: preprocessing, fit the model and grid search for the best parameters.

Similar approach was use to address the task that identify float windows from non-float windows. This time, not all observation was used. Only observations with label 1, 2, 3 was used, since the task was focus on classify different types of windows.

Refinement

For SVM, default parameters, {'C': 1, 'gamma': 0.05, 'kernel': 'rbf'} was used at beginning. F1_score for the result is 0.69. Then grid search was used to find optimal parameter. During the grid search, parameter grid was adjusted according to the result to further optimize the parameters. The final parameter for SVM were {'C': 90, 'gamma': 0.05, 'kernel': 'rbf'}. This parameter setting was able to get

f1_score of 0.717, which slightly better than the default. C is the penalty term, which can reduce the variance. Gamma is the parameter used in the kernel. For SVM the most important parameter is the kernel. If I change kernel to "linear", the f1_score decrease to 0.63.

For KNN, similar approach was used, and the final parameter setting was {'n_neighbors': 3, 'p': 1}. The best f1_score was 0.715 similar to the SVM. The parameter p determine which distance measure is used; gird search choose from Manhattan distance and Euclidean distance, where p equals to 1 and 2 respectively.

For Random forests, the optimal parameter setting was {'criterion': 'gini', 'n_estimators': 200, "max_depth":15, "max_features": "auto" }. The grid search tend to choose highest n_estimators, but the f1_score didn't' increase much. I choose 200 for a relative same performance. Both n_estimators and max_depth have significant impact on the performance. Max_depth determine how deep each tree learner can be. On an extreme, setting max_depth to 2 , the f1_score decease to 0.53. However, if I keep max_depth as 15, decrease the n_estiamte to 10, the f1_score is still as high as 0.73. The best f1_score was 0.797, which is the best of three algorithms.

For binary classification phase. The optimal model from multi-class phase was use. For both task, f1_score was reasonable high. Therefore, there are not much room for improvement. No grid search was used in this phase. The detail results will be reported in next section.

Results

Model Evaluation and Validation

The parameters for each optimal model, and it f1_score was provided in the table below. The f1_score was obtain by taking average of 5-fold cross validation.

Model	Optimal parameters	F1_score
SVM	'C': 90, 'gamma': 0.05, 'kernel': 'rbf'	0.717
KNN	'n_neighbors': 3, 'p': 1	0.715
Random Forests	'criterion': 'gini', 'n_estimators': 200, max_depth = 15, max_feature ="auto"	0.791

Since the f1_score is the harmonic mean of the recall and precision, the classification performance of the model is reasonable good, given what we have to train the model. Following is the confusion matrix on the test set, where we can inspect the model performance more concretely.

For KNN:

		Predicted value					
True value	1	18	4	1	0	0	0
	2	2	12	0	2	0	0
	3	5	2	0	0	0	0
	5	0	0	0	1	0	0
	6	0	0	0	0	1	0
	7	0	0	0	0	0	6

For SVM:

		Predicted value					
True value	1	17	3	2	0	0	1
	2	3	11	0	1	1	0
	3	1	3	3	0	0	0
	5	0	0	0	1	0	0
	6	0	0	0	0	1	0
	7	0	0	0	0	0	6

For random forests:

		Predicted value					
True value	1	21	1	1	0	0	0
	2	1	13	0	0	1	1
	3	4	2	1	0	0	0
	5	0	0	0	1	0	0
	6	0	0	0	0	1	0
	7	0	0	0	0	0	6

We can see that random forests make least mistakes on the test.

For the binary case, I just use the same parameter setting. And the result are presented below:

- Whether windows:

Model	F1_score
SVM	0.944
KNN	0.903
Random Forests	0.963

Again, the random forests has the best performance, although KNN has the same performance on test set. (see confusion matrix below). We can visualize the result by a ROC plot based on test set (for random forests only) and confusion matrix.

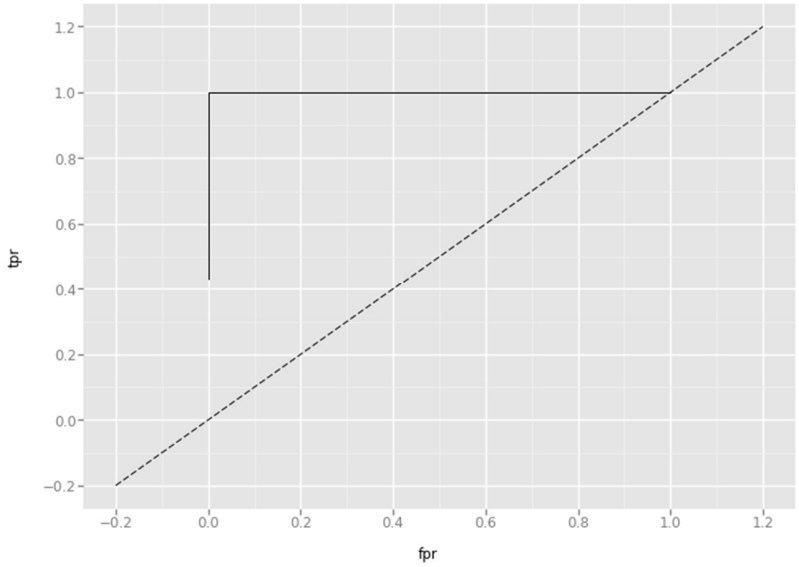

Figure 5 ROC

We can see that the ROC curve is ideal; the error is too small to be drawn in the graph.

We can compute the confusion matrix for each algorithms, using the same test set,(0 stand for non-window)

Confusion Matrix predicted							
		SVM		Random Forests		KNN	
True Value	0	8	0	8	0	8	0
	1	3	43	2	44	2	44

To compare with benchmark, I use Random Forest to compute the confusion matrix using entire dataset as test set.

		Predicted Value	
True Value	0	51	0
	1	2	161

We can see our result is much better than that of original paper, since our model only make 2 mistake: classify 2 windows samples as non-windows samples.

- Whether float windows:

Next, we use our models to test whether a windows is float processed. The table below shows the f1_score of each model (mean of the 5-fold cross validation is used).

Model	F1_score
SVM	0.747
KNN	0.775
Random Forests	0.843

We can see this is much more difficult task. Let's see the confusion matrix,(0 stand for non-float):

Confusion Matrix predicted							
		SVM		Random Forests		KNN	
True Value	0	17	4	19	2	17	4
	1	3	17	2	18	2	18

The random forests still outperform other two algorithms. Let us see the confusion matrix with entire dataset as test set in order to compare with the benchmark.

		Predicted Value	
True Value	0	74	2
	1	2	85

We can see that the random forests model only make 4 mistakes, much better than the benchmark 28 mistakes. In addition, worth noticed that, this result is misleading due to using training data set as testing set. Nevertheless, the results on test set is also reasonably good.

Conclusion

This project use public available glass sample dataset to develop a model to identify different types of glass. The results show that random forests has best performance in various tests, which is not very surprising that ensemble approach usually has better performance. The dataset has some issue like imbalanced labels, certain categories only have several observation, and total size of the dataset is quite small, this may cause some problem in developing model. Cross validation is used to measure the performance in order to compensate the lack of data. We can see that the model perform quite well in identifying windows from non-windows. It is reasonable to believe that the model will perform decent for new data. However, in multi-classification phase, certain type of glass, like tableware, has too few observations to make the model convincing that it will perform well for new data. Nevertheless, this project has some meaningful exploration in glass identification; given enough data, the similar researches will provide more insights into to crime investigations and forensic science.

Improvement

I think the most significant issue that may cause harm to the results is lack of data. The dataset only has 214 observations and has significant imbalanced label problem. The model may easily over fit to the entire dataset. However, data of its kind is not easily obtained via public sources. If extra data can be obtained, the model will be improved for sure in both reliability and accuracy.

Reference

Ian W. Evett and Ernest J. Spiehler. Rule Induction in Forensic Science. Central Research Establishment. Home Office Forensic Science Service. Aldermaston, Reading, Berkshire RG7 4PN