1. Accuracy Analysis

In assignment 3, I used python as the scripting language with modules including Scikit-Learn, Imbalanced-Learn, and NLTK. The scripting codes are also available within the submission, more details about the accuracy are provided by the README.md or by running the scripting codes.

First, the average accuracy of ten-folding cross-validation on each classifier without tuning hyper-parameters are: 0.794545 - 0.805455 (Decision Tree), 0.829091 - 0.830909 (Random Forest), 0.816364 (Logistic Regression), and 0.830909 (SVM). Based on the repeatedly tested results, SVM is considered the classification model with the best performance in this situation, and Decision Tree and Random Forest may deliver 2 slightly different accuracy results, which I think can be attributed to the diverse shapes of the trees generated.

Second, the application of SMOTE doesn’t boost the accuracies of classifiers, it results in a severe decline instead in most cases (Decision Tree 0.630909, Random Forest 0.680000, Logistic Regression 0.665455). I attempted to attribute this phenomenon to the meaninglessness of the string features generated/resampled by the SMOTE, so I used the Matplotlib module to create the scatter diagrams for analysis. Figure 1 shows the distribution of the seventh feature values (data structure: string) over the label in the first fold of validation, and SMOTE seems like modifying the seventh feature of both labels’ classes even if I set the sampling strategy to minority.

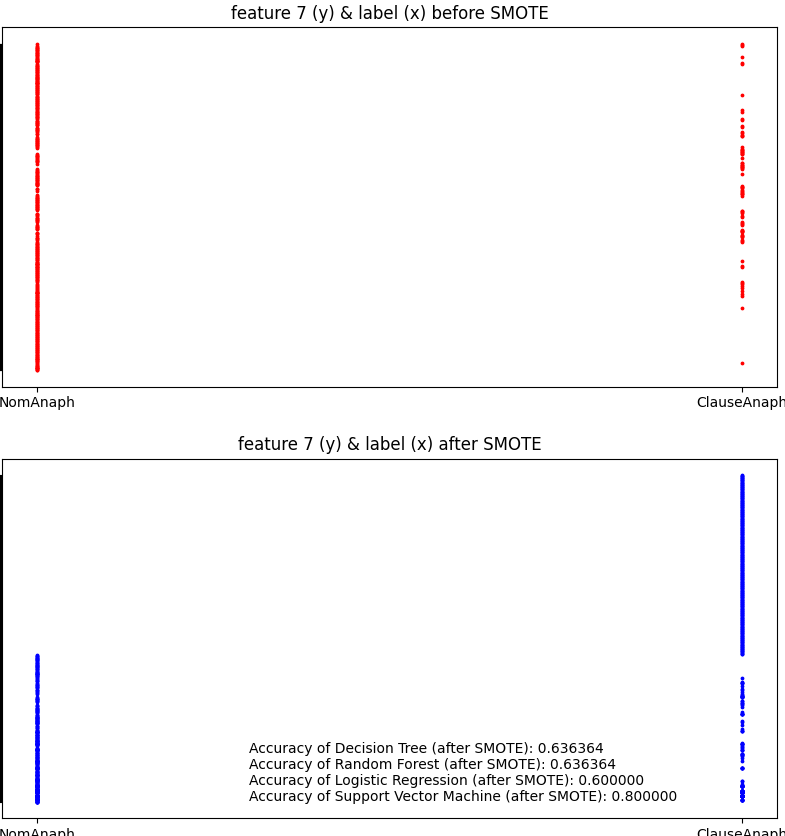


Figure 1, Distribution of feature 7 over the label

Thus, I created more diagrams to show the distribution of the seventh feature values over the first feature values (the position of the word “it” in a sentence), figure 2 shows a scatter diagram of the distributions before and after the SMOTE in the first fold of validation, we can clearly see the indiscriminate modification by SMOTE on the raw dataset (clear shifts of dots). Thus, my hypothesis is correct, and I can conclude that SMOTE technique is not suitable for text-based features.

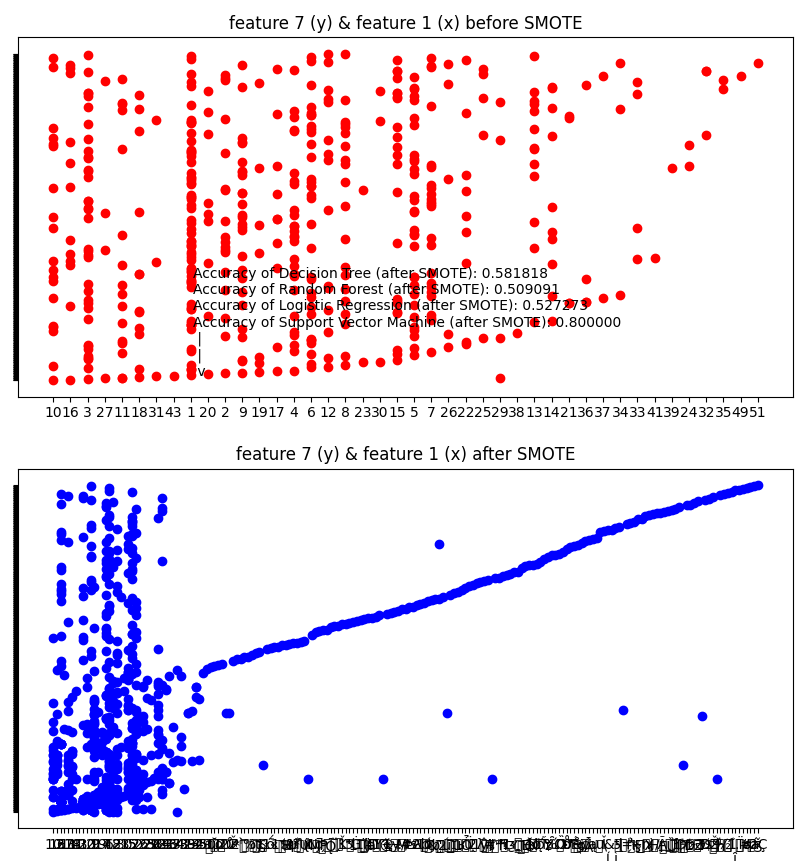


Figure 2, Distribution of feature 7 over the feature 1

However, the SVM classifier’s accuracy remains after SMOTE is applied, this also shows the advantageous characteristic of SVM, which copes with disruptive features well.

Third, I tuned the hyper-parameters of classifiers with the values that fall in the ranges provided by the assignment description and found that 1. Tuning hyper-parameters leads to the boost of average accuracy for Decision Tree (0.830909), Random Forest (0.830909), and Logistic Regression (0. 830909), 2. The different values in these ranges merely result in trivial changes in terms of accuracy in each fold of validation.

1. Feature Ranking

Output of my script:

C:\Users\Heng\Desktop\App.-of-NLP-ML-in-SE\assignment 3>python information\_gain.py

Dataset Loaded.

Features and Labels Extracted

H(C) is computed as: 0.665165

Feature 7's information gain is 0.636284

Feature 18's information gain is 0.614845

Feature 2's information gain is 0.103464

Feature 1's information gain is 0.070840

Feature 4's information gain is 0.039073

Feature 15's information gain is 0.034273

Feature 16's information gain is 0.017386

Feature 5's information gain is 0.017224

Feature 3's information gain is 0.016757

Feature 10's information gain is 0.012787

Feature 9's information gain is 0.010067

Feature 11's information gain is 0.008989

Feature 20's information gain is 0.008032

Feature 19's information gain is 0.002490

Feature 6's information gain is 0.001491

Feature 13's information gain is 0.001428

Feature 14's information gain is 0.001111

Feature 8's information gain is 0.000496

Feature 12's information gain is 0.000143

Feature 17's information gain is 0.000000

Scikit-learn doesn’t implement the functions to calculate information gain, so I had to write my own code to do so. I firstly used the following formula to calculate the total entropy:

H(C) = )

And then I used the following formula to calculate the information gain of each feature:

Info(T) = H(C) - H(C|T)

= H(C) - )

= H(C) - , where T is the possible values of a specific feature,

t is the possible labels of that feature with value T,

spilt(T) is the quantity of that feature instances with value T divided by the quantity of all instances

p(NA) is the possibility of that feature instance with value T has label NomAnaph

p(CA) is the possibility of that feature instance with value T has label ClauseAnaph

1. Lessons Learned
2. Random Forest needs more time to finish training because it internally contains many Decision Tree classifiers needed to be trained.
3. SVM is the best classification model among the four in terms of accuracy, capability to cope with disruptive feature values (e.g., strings generated by SMOTE), and stability.
4. I have understood the working mechanism of SMOTE, which resamples the dataset and generates more data instances of minor labels in order to balance the data.
5. SMOTE can bring a strong negative impact (about 0.2 lower accuracies in my case) on the classifiers taking the text as feature(s).
6. Reasonably tuning hyper-parameters can boost the accuracies of classifiers, especially for Decision Tree, Random Forest, and Logistic Regression
7. I am more familiar with the calculations of entropy and information gain through the implementation of them.