Binary Classification on Imbalanced Dataset

Yifan Bai **Bilal Chabane**

Maxime Daigle

Abstract

According to no free lunch theorem, there is no single algorithm that works best for every problems. In this project, we seek to investigate whether one of the three selected algorithms works best for a single type of problem, binary classification on small dataset with imbalanced classes, and compare the performances as well as advantages/disadvantages of each algorithm on two similarly styled datasets.

1 Introduction

Three binary classifiers were selected: SVM, Random forest, and Multilayer perceptron. Those classifiers were selected because they are among the most common used binary classifiers and, at the same time, they are notably different. SVM focuses on separating the classes with the largest margin. A perceptron separates the data in a similar manner except that it doesn't take into account the margin. However, a multilayer perceptron has the advantage of having a wide range of capacity. Finally, random forest, an ensemble method using decision trees, is particularly different with its decision boundary based on axis-orthogonal regions.

To test their performances, two datasets were chosen: Bank telemarketing and Income prediction. Those datasets were selected because they are similar imbalanced datasets. Furthermore, bank telemarketing datasets draw test samples from the master datasets which would be sufficient to only have cross validation representing the size of the test set (10:1), whereas income prediction dataset is given as a split of the master dataset, therefore we perform tests in addition to cross validation.

The bank telemarketing dataset contains personal information on bank clients (age, marital status, education, etc.) and the task is to predict whether the client will subscribe to a term deposit or not. The dataset contains 45,211 clients with 16 features for each. The classes are imbalanced (88.3% "no" v.s. 11.7% "yes").

Similarly, the income prediction dataset also contains personal information (age, marital status, education, etc.) and the task is to predict whether the person makes over $50,000 a year or not. The dataset contains 45,222 samples with 14 features. The classes are imbalanced (75.22% making less v.s. 24.78% making more).

2 Experiments

The datasets were cleaned, but no feature engineering was done. Each algorithm had their hyperparameters tuned (See Section A.1 in Appendix) and their best performance selected.

We can see that multilayer perceptron and random forest obtain better results than SVM. Between multilayer perceptron and random forest,the accuracy is almost the same for the bank data, but random forest obtains an accuracy 5% higher than multilayer perceptron for the income data. Therefore, even if the accuracies are close, random forest has a better performance on both datasets.

Table 1: Best performance of each algorithm

Classifier	Bank Accuracy	Income Accuracy
SVM	89.12	76.12
Random forest	**90.54**	**84.76**
Multilayer perceptron	90.08	79.72

3 Discussion

3.1 Algorithm theoretically more likely to overfit

Among the three algorithms, multilayer perceptron is theoretically more likely to overfit. Multilayer perceptron is one type of neural network, and theoretically it is able to approximate any type of functions. Therefore, it is more likely to fit "noise" in the training data. A neural network overfitting will get large test error comparatively to the trainning error. In practice, however, we didn't have any overfitting problem with any of the algorithms.

3.2 Robustness of algorithms' performance

There are multiple things leading to the differences among algorithms' performance: training data, hyperparameters, training strategies. We can see that random forest can provide the most robust performance among different hyperparameters, and multilayer perceptron has the least robust performance among different hyperparameters - it could perform better than random forest and it could also perform worse than SVM. The reason is that multilayer perceptron has a wide range of capacity. Those levels of robustness for the algorithms' performance can be seen, among others way, with the tuning of hyperparameters on the bank dataset.

Table 2: Performance on bank data before and after tuning the hyperparameters

Classifier	Accuracy before	Accuracy after	Improvement
SVM	88.29	89.12	0.83
Random forest	89.98	90.54	0.56
Multilayer perceptron	88.37	90.08	1.71

Here, tuning the hyperparameters improves the accuracy by 0.56% for random forest. However, it improves the accuracy by 1.71% for the multilayer perceptron. It is more than three times the improvement of random forest.

If we look more closely to the tuning of some random forest's hyperparameters, we can indeed see that there is no particular hyperparameter that drastically worsen or improve the performance. In certain cases, greedily increasing values of certain parameters, for instance, maximum depth, induces overfitting and decreases accuracy. THese observations highlight the importance of tuning to find the best set of hyperparameters, which could be done by grid search.

In Figure 1, the biggest impact of modifying the number of estimators in random forest is a difference of 0.21% with the accuracies at number of estimators = 50 and number of estimators = 100 (90.32% and 90.53%).

In Figure 2, the biggest impact of modifying the maximum depth is a difference of 0.74% with the accuracies at maximum depth = 5 and maximum depth = 16 (89.35% and 90.09%).

In Figure 3, validation accuracy and test accuracy follow the same patterns, with overfitting peaking at 1.62% for maximum depth and 0.98% for number of estimators; the highest validation/test accuracies occur when maximum depth = 16 and number of estimators = 500 for validation and number of estimators = 200 for test (85.77%/85.20% and 85.58%/84.70%).

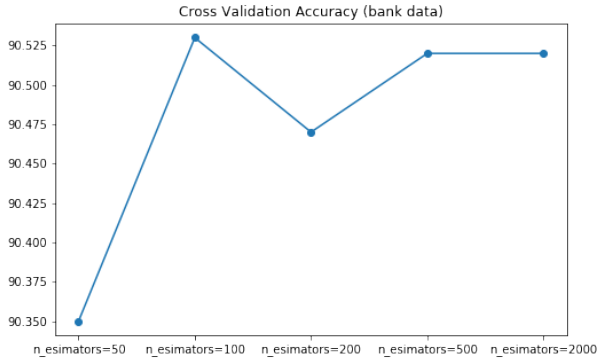

Figure 1: Effect of number of estimators on cross validation accuracy

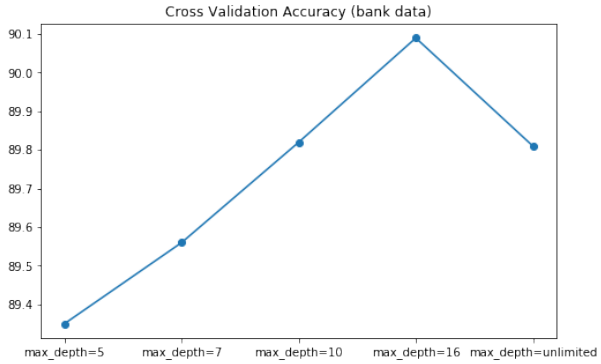

Figure 2: Effect of maximum depth on cross validation accuracy

3.3 Data distribution

Many algorithms are sensitive to the way the data is distributed and perform better when features are on similar scale and/or relatively normally distributed. In our case, the data distribution would mostly affect the SVM and the multilayer perceptron.

For example, SVM with RBF kernel assumes that the features are with zero mean and have variance in the same order. However, it is not the case with our datasets.

Table 3: Mean and Standard deviation of features from each datasets

	Bank		Income	
	Age	Balance	Age	Fnlwgt
Mean	40.94	1362.27	38.58	1.897e+05
Standard deviation	10.62	3044.77	13.64	1.056e+05

Therefore, standardizing could improve the performance of SVM with RBF kernel which is already the SVM with the best performance (See Table 5 and Table 8). Furthermore, big difference in range between features can also deteriorate the performance and both datasets have this characteristic.

3.4 Datasets used and real-world problems insights

When dealing with real-world problems, the data available is often not ideal. In our case, we can see that random forest and multilayer perceptron can generally perform better than SVM, this means the data is not easily separable like a linearly separable dataset. The decision boundary between two classes could be very complex. This is very common in real-world problems. Also, in real-world problem, it is not surprising to have imbalanced data. Additionally, there is no strong features that are especially helpful by themselves for the classification which is something that would also not be

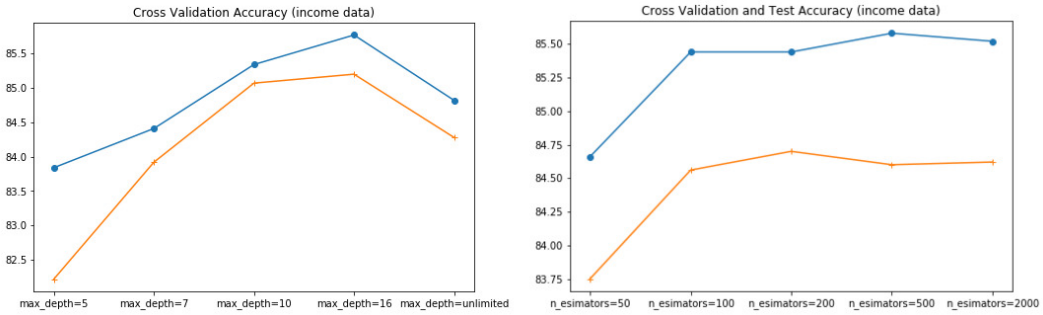

Figure 3: Effect of maximum depth/Number of estimators on validation vs test accuracy

Table 4: Minimum and Maximum of features from each datasets

	Bank		Income	
	Age	Balance	Age	Fnlwgt
Minimum	18	-8019	17	1.23e+04
Maximum	95	1.02e+05	90	1.48e+06

surprising in real-world problems. However, the datasets used, contrarily to real-world problems, are very cleaned data, where in real-world we will usually have raw data with bad and missing values.

3.5 Recommendations for facing similar real-world problems

For similar real-world problems, we will recommend to use random forest as the baseline algorithm, usually the default settings provided by sklearn is a good start. As we can see random forest can give better results than SVM and multilayer perceptron (in some cases). Also, random forest can be trained very fast on this size of the data. In addition, random forest can output feature importance which will be helpful for feature engineering and model evaluation.

4 Conclusion

In general, random forest seems the best choice for both of the imbalanced datasets. It obtains the best results (even if by not much), it is the more robust to change in its hyperparameters, it is trained very fast, and it has a good interpretability. However, there is multiple topics that would be interesting to explore for further work.

Firstly, it would be interesting to try to standardize the inputs. Random forest is not sensitive to difference of scale in the inputs. However, standardizing could help the multilayer perceptron and the SVM. Therefore, it could be possible that random forest wouldn't be the algorithm obtaining the best performance on both datasets with some feature engineering.

Secondly, it would be interesting to determine which features contribute the most to the classification. Feature selection could change the performance of the algorithms. Some algorithms could be more sensitive to the noise contained in unhelpful features.

Thirdly, it would be interesting to look which techniques to deal with imbalanced data work the best and to look how the performance of the different algorithms are affected by those techniques.

Finally, it would be interesting to try more algorithms and to use a bigger variety of similar imbalanced datasets. It could allow to investigate in more depth the general sensitivity to imbalanced classes of those algorithms.

Acknowledgments

All team members contributed equally for each component of the project, which includes but is not limited to: defining the problem,developing the methodology, coding the solution, performing the data analysis, writing the report.

A Appendix

A.1 Top 3 best results for each algorithm

Table 5: **SVM** performance on **bank** data according to the hyperparameters

C	Kernel	Gamma	Accuracy
10	rbf	0.0001	89.12
10	rbf	scale	89.05
5	rbf	scale	88.84

Table 6: **Random forest** performance on **bank** data according to the hyperparameters

Number of estimators	Criterion	Max depth	Min samples split	Min samples leaf	Accuracy
1500	gini	None	2	1	90.51
1000	gini	None	2	1	90.54
500	gini	None	2	1	90.52

Table 7: **Multilayer perceptron** performance on **bank** data according to the hyperparameters

Structure	batch size	Accuracy
100, 100	256	89.87
200, 100	256	89.99
200, 200	256	90.08

Table 8: **SVM** performance on **income** data according to the hyperparameters

C	Kernel	Gamma	Accuracy
10	rbf	scale	75.88
1	rbf	scale	75.05
10	rbf	0.0001	76.12

Table 9: **Random forest** performance on **income** data according to the hyperparameters

Number of estimators	Criterion	Max depth	Min samples split	Min samples leaf	Accuracy
1500	gini	None	2	1	84.66
1500	gini	None	10	1	84.73
1500	gini	None	20	5	84.76

Table 10: **Multilayer perceptron** performance on **income** data according to the hyperparameters

Structure	batch size	Accuracy
100, 100	256	79.54
200, 100	256	79.68
200, 200	256	79.72