UCLouvain

Louvain School Of Engineering

LINFO2262: Machine Learning: classification and evaluation: Final Project

Author:
Alexandra Onciul - 54212000

Professor:
Dupont Pierre

Abstract

In this project, we were confronted with a prediction task inspired by a biomedical research topic. We had to build the best possible classification model and predict the actual classification performance.

1 Design Choices

I chose to make it undeterministic by not setting the random states.

1.1 Preprocessing

Firstly, I normalized the data. Applying the decribe() method, I got a closer look at how the data looked. I decided to apply these:

- **Columns 'event' and 'node':** These columns contain boolean data. We converted the values when True to 1 and when False to 0.
- **Column 'grade':** This column was processed using one-hot encoding, which transforms the categorical grade values into multiple binary columns, each representing one category. Each resulting binary column was then converted to integer type.
- **Other columns:** All other columns were normalized using the `StandardScaler` from the `scikit-learn` library, which standardizes features by removing the mean and scaling to unit variance. This is particularly important to prevent attributes with initially larger ranges from outweighing attributes with smaller ranges. But the scaler was reinitialized for the time, sizeTum and age column in order to be more meticulous and have sturdier values as BCR.

Furthermore, the label normalization was conducted by replacing string labels with integer codes to facilitate processing in machine learning models. Specifically, breast cancer receptor statuses were encoded as follows:

$$
\begin{aligned}
\text{'ER+/HER2-'} &\rightarrow 0\\
\text{'ER-/HER2-'} &\rightarrow 1\\
\text{'HER2+'} &\rightarrow 2
\end{aligned}
$$

As I observed that the dataset is consenquently imbalanced so I applied the **SMOTE** algorithm in order to duplicate lesser represented classes and reduce overfitting due to the training set. After multiple feature selection algorithms tries (such as PCA and SelectKbest), I applied an **SelectFromModel** with as estimator an **SVC** with a linear kernel and tuned parameters:

- **kernel** = 'linear'
- **gamma** = 0.001
- **C** = 0.01

SelectFromModel:

- **selector** = SVC
- **threshold** = '1.4*mean'

1.2 Algorithm

After a lot of computations, and keeping in mind the increased overfitting possibility of certains algorithms. I found the best model (of the tested parameters) to perform on this task as a **Linear Regression** with following parameters:

- **multi_class** = 'multinomial'
- **solver** = 'saga'
- **max_iter** = 1000
- **tol** = 1×10^{-1}
- **penalty** = 'l1'
- **C** = 0.5
- **fit_intercept** = True

We can see that the training accuracy is quite high and the test accuracy stabilizes around 85 percent.

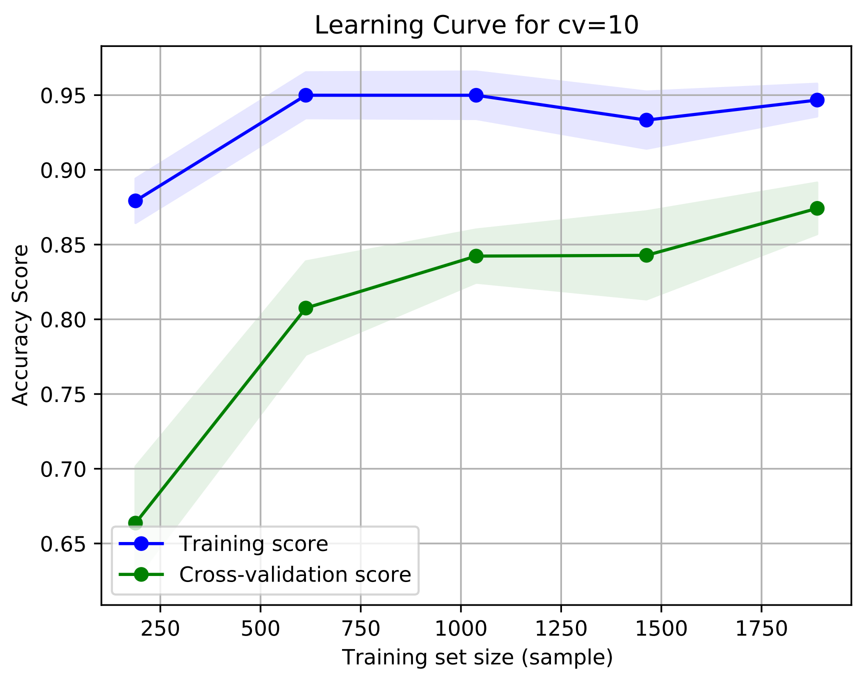

Figure 1: Learning Rate for cv=10

2 Evaluation

To compare the models, I used GridSearch and RandomGridSearch with cross validation with 10 folds with as scoring function balanced_accuracy of Sklearn.

To evaluate the best model, I did multiple iterations where I splitted the data ramdomly with a test set of 20 percent of the initial training set. BCR is particularly useful for inbalanced datasets. I computed the confidence interval as well.

3 Methodology

In this section, we describe the methodology followed for conducting the machine learning experiments. I first did some research in order to see if I can get some information about gene proccessing and classifiers problems and how people managed to overcome this task. I didn't investigate the complicated models as the time is limited and high computional time. I implemented a pipeline in order to test the parameters more easily. This is important as the scaling needs to be done just after the splits so that it doesn't know the other values yet.

3.1 Initial Model Testing

Initially, each model (decision tree classifier, random forest, bagging, support vector machine (SVM), and deep learning) was evaluated using codes originally written for other phases. I tried first without Principal Component analysis, Feature Selection nor Normalization, in order to get a first idea. Then I added Normalization and then tried with either SelectKBest, PCA or SelectFromModel. I tried to tune and investigate with RandomSearchCv how the preprocessing and classifier interacted.

3.2 Parameter Tuning

After the initial testing, parameter selection for each model was conducted. The goal was to identify the combination of parameter values that produced the best performance for each model with code previously written a lot was done through GridSearchCv. But I also experimented with different values for parameters of preprocessing.

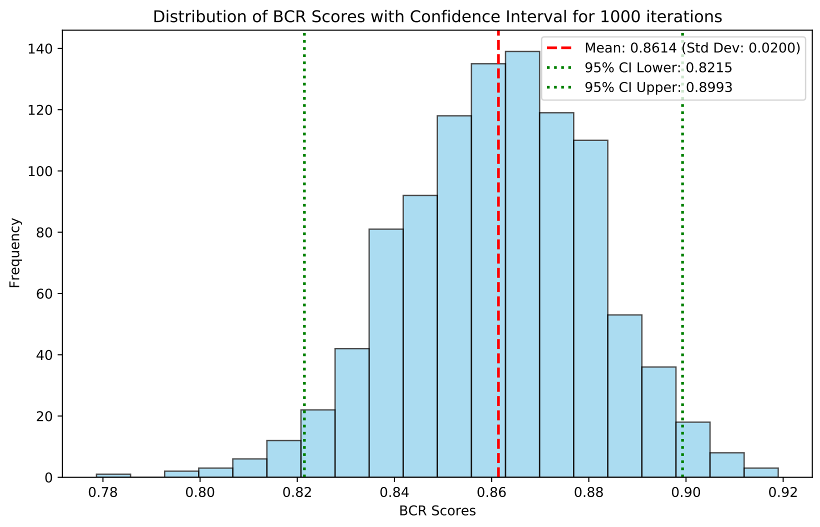

Figure 2: Confidence interval of BCR scores for alpha=0.05

3.3 Focus on Specific Models

The focus was primarily on three models: deep learning, random forest, and support vector machine. As I observed the SVM behaves really well even without any preprocess I conducted other researches and penalized LogisticRegression came up and gave me really good BCR scores with less variability. I chose thus the l1 Lasso Regression[1].

3.4 Intensive Tuning

These selected models underwent further tuning to refine their performance as much as possible. This involved fine-tuning the parameter values to achieve optimal results with GridSearchCv.

4 Conclusion

This task was complicated but really interesting as there were a lot of possible pipelines. This illustrates the complexity of the real world. I have a quite large confidence interval.

References

Zhu, Ji and Trevor Hastie (July 2004). “Classification of gene microarrays by penalized logistic regression”. In: *Biostatistics* 5.3.

[1]Zhu and Hastie 2004.