Big Home Assignment

Dzheykhun Kasumov

December 6, 2023

Introduction

In this paper we will introduce the thorough analysis and comparison of six machine learning algorithms on three real-world datasets taken from the open source ML competitions website Kaggle. The aim of this paper is to compare the performance of the baseline Lazy Classification FCA models (binarized and "patternized") with the classical ML classification algorithms (kNN, Logistic Regression, Random Forest and XGBoost). Project Github repository for the code details: **https://github.com/killiganni/LazyFCA_OSDA**. For the analysis, we took three real-world datasets:

1. Loan Default Prediction Dataset.

   The link: https://www.kaggle.com/datasets/nikhil1e9/loan-default

2. Water Quality Dataset.

   The link: https://www.kaggle.com/datasets/adityakadiwal/water-potability

3. Stroke Prediction Dataset.

   The link: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

These datasets are stored in the special folder in the project Github repository. Overall, this repository consists of the following files:

1. Report. The project report is stored here.

2. Data. The project datasets (Loan default, Water quality and Stroke prediction) are stored here.

3. Code. The project Python code is stored here.

Dataset 1: Loan Default Prediction.

As it was mentioned, in this project we'll analyze three datasets, and the first one is connected with Loan Default prediction. Loan default prediction is crucial for several reasons, especially for financial institutions and lenders. It is important for maintaining financial stability, reducing risks, ensuring responsible lending practices, and fostering a healthy economic environment for both lenders and borrowers.

Before applying the abovementioned ML algorithms the Exploratory Data Analysis was performed, and some necessary steps were conducted. Firstly, it should be mentioned that dealing with 255000 objects (which is the number of entries of Loan Default dataset) is an infeasible task for the LazyClassification methods. So, in order for LazyClassification methods to deal with the Loan Default dataset, we randomly selected 5000 entries of this dataset. All other (classical) ML models were trained on the full version of the dataset.

For all of the models, categorical variables were transformed into sets of dummy variables using pd.get_dummies function. For the Binarized LazyFCA algorithm we also binarized all of the numeric features using some of the rules: for instance, for the Age variable we splitted data into four groups: Ages of 18-25 (Youth), 25-35 (Early Adulthood), 35-50 (Later Adulthood) and 50-69 (Advanced Age). For the division of all of the other variables we would need more of the domain knowledge, which in our case is unavailable. So, in order to make the division of other variables, we applied a simple rule: divide data by quantiles (25%, 50%, 75% percentiles).

To generate the predictions of LazyFCA methods we used the most standard method of prediction (the first one out of three), as it (surprisingly) showed the best results.

In order to let logistic regression and other models account for non-linearities we also augmented the feature space by adding quadratic and cross features of degree 2.

All statistical methods were assessed using statistical classification metrics: Accuracy and F-1 Score, measured in percents.

All of the hyperparameters of classical ML models were chosen using the GridSearch technique: for kNN algorithm, we chose n_neighbors=20 (it appeared to give the highest cross-validation score when performing the GridSearchCV). For Logistic Regression, the budget hyperparameter equal to 1 showed the best results, while for Random Forest classifier we chose 500 trees each of the

size-4 depth. XGBoost hyperparameters are as follows: learning_rate = 0.1, max_depth=7, n_estimators=180.

	Accuracy	F-1 Score
LazyFCA (binarized)	58 %	23.4 %
LazyFCA (patternized)	77 %	29.6 %
kNN	80.6 %	20 %
Logistic Regression	70 %	32.7 %
Random Forest	70.4 %	35.1 %
XGBoost	80 %	35.6 %

We may observe a lot of insides from the above table. Generally, it may be noted that though showing similar accuracy (as in the case of LazyFCA patternized model), the baseline LazyFCA models show significantly lower F1 scores compared to other classical ML algorithms, which may indicate that these baseline algorithms are somewhat inferior to the classical models we often use in other ML research. All of the inferences will be showed in the conclusion section.

Dataset 2: Water Potability Prediction.

The second dataset analyzed is connected with water potability prediction. Water potability prediction holds significant importance due to its direct impact on public health and well-being. It involves the assessment and prediction of whether water is safe for human consumption based on various chemical, physical, and microbial parameters.

Overall, the preprocessing technique is quite similar to the one in the previous dataset. However, now we deal with just 3276 entries, so there is no need to randomly select objects for the LazyFCA methods. Moreover, now we encounter the problem of NA values, which is solved using the fillna pandas function, which fills NA values with the mean values of the corresponding column. Still, we binarize the numeric data for the Binarized LazyFCA algorithm, and we generate polynomial features of degree 2 (larger degree leads to extremely high computational complexity) to account for non-linearities, which may be captured by the classical ML algorithms.

All statistical methods were still assessed using statistical classification metrics: Accuracy and F-1 Score, measured in percents.

	Accuracy	F-1 Score
LazyFCA (binarized)	56 %	42 %
LazyFCA (patternized)	61 %	5 %
kNN	61 %	18 %
Logistic Regression	56.7 %	35 %
Random Forest	68.9 %	58.8 %
XGBoost	65.1 %	59 %

This time the hyperparameters are as follows:

1. kNN algorithm: n_neighbors=35.
2. Logistic Regression: budget parameter C = 0.5.
3. Random Forest Classifier: max_depth: 8, n_estimators: 500.
4. XGBoost: learning_rate: 0.1, max_depth: 6, n_estimators: 100.

Interestingly, we still observe the same result: though the accuracy scores are quite similar, the F1 scores are indeed different for the LazyFCA and classical ML algorithms. Unsurprisingly, the best results are still shown by the gradient boosting technique called XGBoost.

Dataset 3: Stroke Prediction.

The final dataset analyzed deals with stroke prediction. According to the World Health Organization (WHO) stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. This dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relavant information about the patient.

The preprocessing and algorithmic schemes in this case are essentially the same as in the previous two datasets: we preprocess data via dealing with NA values, dealing with the issue of categorical & numerical data binarization, introduction of second-degree polynomial features and etc.

All statistical methods were still assessed using statistical classification metrics: Accuracy and F-1 Score, measured in percents.

	Accuracy	F-1 Score
LazyFCA (binarized)	73 %	22.9 %
LazyFCA (patternized)	81 %	23.7 %
kNN	82.2 %	10.8 %
Logistic Regression	86.9 %	27.3 %
Random Forest	75.2 %	26.7 %
XGBoost	90.2 %	32 %

For the final time, the hyperparameters are as follows:

1. kNN algorithm: n_neighbors=15.
2. Logistic Regression: budget parameter C = 0.2.
3. Random Forest Classifier: max_depth: 6, n_estimators: 200.
4. XGBoost: learning_rate: 0.1, max_depth: 6, n_estimators: 180.

Once again, the table above shows us the very same results we obtained in the previous two cases: LazyFCA baseline models are inferior to the classical ML algorithms (as of F1 scores), and the best results are still shown by the special technique called XGBoost.

Conclusion

Overall, we may observe that almost all of the analyzed classical ML tools (Logistic Regression, Random Forest Classifier and XGBoost) show better results compared to both baseline LazyFCA models and the k-Nearest Neighbors model. And we point out that unsurprisingly the best results as of accuracy and F1 scores are shown by XGBoost gradient boosting algorithm. Obviously, it should be stated that with the increased computational power we could get more benefits from the GridSearchCV technique via analyzing more of the possible hyperparameters combinations. Moreover, increased computational power would allow us to analyze the whole dataset in the case of Loan Default prediction to fully "feed"it to the LazyFCA algorithms. Generally, it means that the following research showed a lot of useful results, though there is still a lot of work that should be done further.