BIG HOMEWORK

ALEXANDER KETSBA

Dec 2023

In this document, I described the application of the LazyFCA algorithm for binary classification using the example of three datasets. The comparison of this algorithm with the usual classifiers is also given: Decision tree, Random forest, XGBoost, Catboost, k-NN, Naive Bayes, logistic regression.

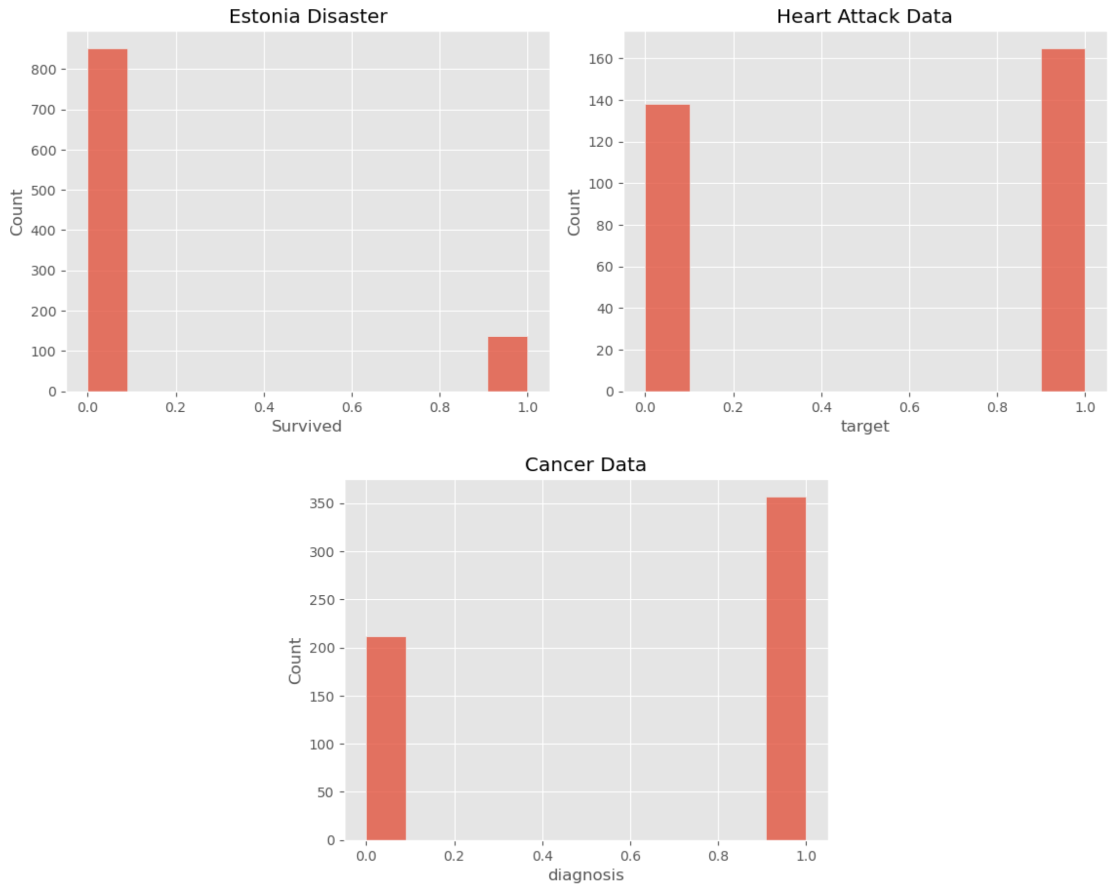
The following datasets were used in the work:

https://www.kaggle.com/datasets/merishnasuwal/breast-cancer-prediction-dataset

https://www.kaggle.com/datasets/christianlillelund/passenger-list-for-the-estonia-ferry-disaster

https://www.kaggle.com/datasets/nareshbhat/health-care-data-set-on-heart-attack-possibility

The target variables are distributed as follows in datasets:



It can be seen that the most unbalanced dataset is Estonia Disaster with a ratio of 86% to 14%.

The dataset with Heart Attack is the most balanced 46% to 54%.

The tools for LazyFCA can be found in the fcalc folder.

A separate notebook has been created for each dataset using models. You can find them along with the files in the GitHub repository: https://github.com/Sandrobus228/BigHW_OSDA

• Cancer_Models.ipynb – notebook using the 1st dataset

• Estonia_Models.ipynb - notebook using the 2nd dataset

• Heart_Models.ipynb - notebook using the 3rd dataset

The feature of the lazy-FCA classification with binary attributes algorithm is to work only with categorical variables. To do this, I binarized continuous variables as follows. For each such variable, I found 10 quantiles [0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1] and I broke down the values by belonging to each of them. Then, using K-Fold, I tested each method for the BinarizedBinaryClassifier algorithm. It is worth noting that the 'standard' and 'standard-support' methods produced predictors [-1, 0, 1]. To combat this and calculate correctly, I replaced the value of '-1' randomly with [0,1]. In the final notebook, I abandoned this idea for a clean result. Divided into 5 splits for K-Fold, the accuracy results for the methods turned out to be as follows:

Estonia Disaster Dataset

```
standard:         mean: 0.4175, max: 0.5, min: 0.3333
standard-support: mean: 0.4176, max: 0.5, min: 0.3333
ratio-support:    mean: 0.5853, max: 0.6087, min: 0.5683
```

Cancer Dataset

```
standard:         mean: 0.8718, max: 0.9, min: 0.8125
standard-support: mean: 0.8668, max: 0.8875, min: 0.8125
ratio-support:    mean: 0.8717, max: 0.9375, min: 0.8227
```

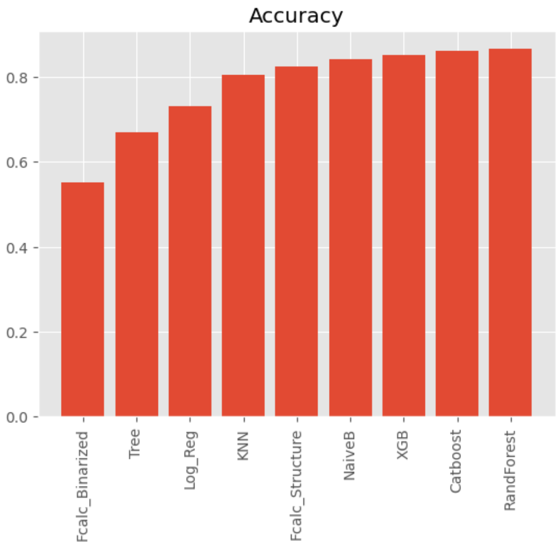
Heart Attack Dataset

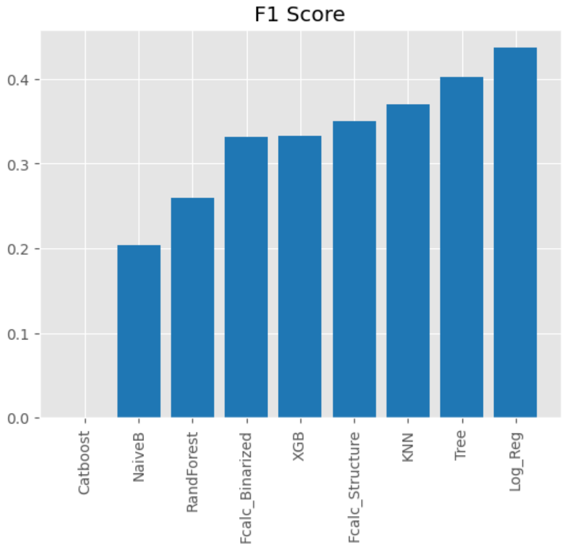
```
standard:         mean: 0.74053, max: 0.8372, min: 0.6428
standard-support: mean: 0.7262, max: 0.7906, min: 0.6428
ratio-support:    mean: 0.7452, max: 0.7674, min: 0.7209
```

It can be seen that in the first dataset, BinarizedBinaryClassifier copes with predictions worse than the "Most frequent class". Most likely, these results were obtained due to a small set of variables.

So, below is a comparison of models based on accuracy and f1 score metrics.

Estonia Disaster Dataset



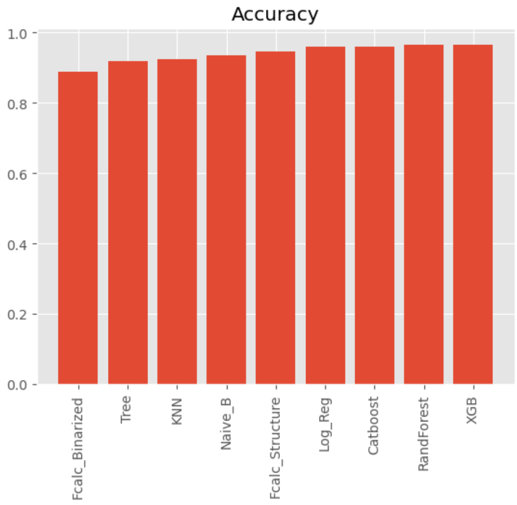


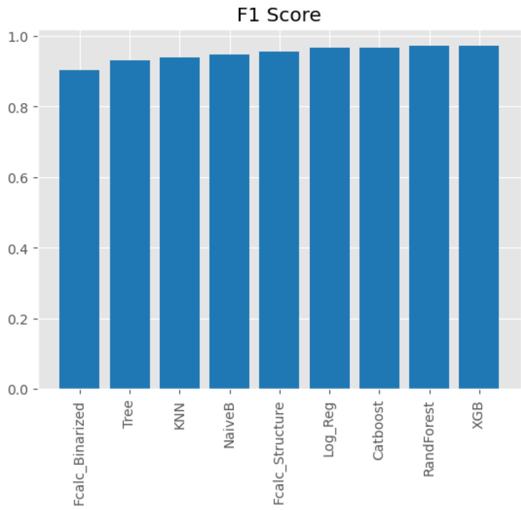

```
Accuracy
----------------------
Fcalc_Binarized: 0.55
Tree:            0.67
Log_Reg:         0.73
KNN:             0.80
Fcalc_Structure: 0.82
NaiveB:          0.84
XGB:             0.85
Catboost:        0.86
RandForest:      0.87
```

```
F1_Score
--------------------
Catboost:        -
NaiveB:          0.20
RandForest:      0.26
Fcalc_Binarized: 0.33
XGB:             0.33
Fcalc_Structure: 0.35
KNN:             0.37
Tree:            0.40
Log_Reg:         0.44
```

It can be seen here that PatternBinaryClassifier does well against the background of other algorithms, but is inferior in both metrics to some of them. Accuracy = 0.82, F1_Score = 0.35. The binary algorithm did not perform so well: Accuracy = 0.55, F1_Score = 0.33.

Cancer Dataset





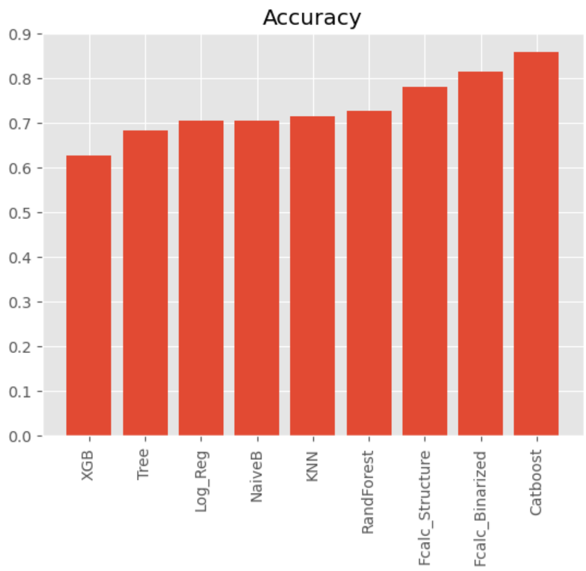
On a more balanced dataset with cancer prediction data, both FCALC algorithms performed well and kept roughly on par with conventional algorithms for both metrics.

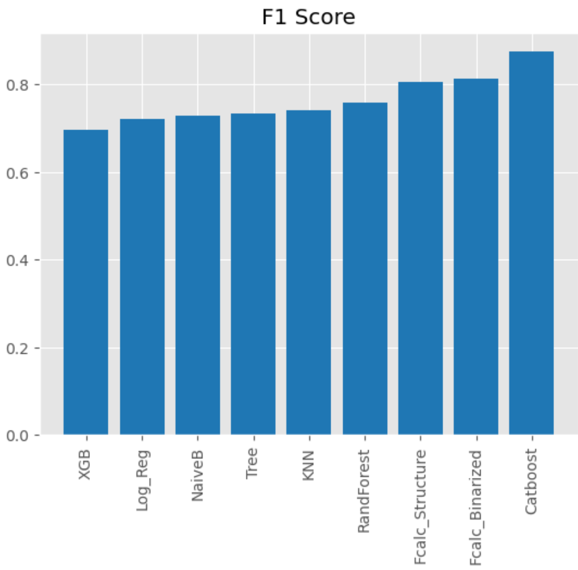
Accuracy	
Fcalc_Binarized:	**0.89**
Tree:	0.92
KNN:	0.92
Naive_B:	0.94
Fcalc_Structure:	**0.95**
Log_Reg:	0.96
Catboost:	0.96
RandForest:	0.96
XGB:	0.96

F1_Score	
Fcalc_Binarized:	**0.9**
Tree:	0.93
KNN:	0.94
NaiveB:	0.95
Fcalc_Structure:	**0.96**
Log_Reg:	0.97
Catboost:	0.97
RandForest:	0.97
XGB:	0.97

Approximately the same can be observed with the Heart Attack data.

Heart Attack Dataset





Accuracy	
XGB:	0.63
Tree:	0.68
Log_Reg:	0.7
NaiveB:	0.7
KNN:	0.71
RandForest:	0.73
Fcalc_Structure:	**0.78**
Fcalc_Binarized:	**0.81**
Catboost:	0.86

F1_Score	
XGB:	0.7
Log_Reg:	0.72
NaiveB:	0.73
Tree:	0.73
KNN:	0.74
RandForest:	0.76
Fcalc_Structure:	**0.8**
Fcalc_Binarized:	**0.81**
Catboost:	0.87

So, we can see that on richer datasets and target-variable balanced datasets, Lazy FCA algorithms perform much better, often even surpassing some familiar and popular classification algorithms.