Big HW

Contents

1 Introduction

This paper aims to conduct a comparative analysis of the effectiveness of prevalent machine learning methods in terms of accuracy and F1 score metrics. The methods under examination include Decision Trees, Random Forest, XGBoost, CatBoost, K-Nearest Neighbors (k-NN), Naive Bayes, Logistic Regression, and LazyFCA. For the implementation of these methods, the libraries FCALC, scikit-learn, CatBoost, and XGBoost were utilized. The entire source code related to this study is accessible at the following URL: https://github.com/dealvost/OSDA-Big-HW. The datasets considered in this study can be found at the following URLs:

1. Heart Failure Clinical Data: https://www.kaggle.com/datasets/andrewmvd/heart-failure-clinical-data
2. Breast Cancer Prediction Dataset: https://www.kaggle.com/datasets/merishnasuwal/breast-cancer-prediction-dataset/
3. Stroke Prediction Dataset: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

2 Datasets Preprocessing

In this study, preprocessing (data preparation and transformation) was conducted on each dataset for subsequent use in machine learning methods.

Each dataset underwent cleansing to remove any rows containing NaN values in any position. Rows with incomplete data for any of the features were eliminated. All columns titled 'id' were removed. Subsequently, 500 rows were randomly selected from each dataset (except for the 'heart_failure' dataset, which initially contained fewer than 500 objects), resulting in the datasets 'breast_cancer_ishodnik' 'heart_failure_ishodnik' and 'heart_stroke_ishodnik'

All preprocessing actions are documented in the file at https://github.com/dealvost/OSDA-Big-HW/blob/main/datasets/Datasets_preparation.ipynb.

2.1 Binarization Process

The datasets underwent binarization according to the following algorithm:

2.1.1 heart_stroke_ishodnik Binarization

- The 'gender' feature was binarized using one-hot encoding, assigning Male = 1, Female = 0.
- The 'ever_married' feature was binarized with one-hot encoding, assigning Yes = 1, No = 0.
- For the 'work_type' feature, which describes 5 unique categories ('Private', 'Self-employed', 'Govt_job', 'Children', 'Never_worked'), each category was added as a separate column, followed by one-hot encoding. The original 'work_type' column was then removed.
- 'Residence_type' was binarized, assigning Urban = 1, Rural = 0.
- 'Smoking_type' binarization was similar to 'work_type', with unique categories ('never smoked', 'smokes', 'formerly smoked') undergoing one-hot encoding. The original 'work_type' column was removed.

The resulting dataset, after binarizing the 'gender', 'ever_married', 'work_type', 'residence_type', and 'smoking_type' features, was named 'heart_stroke_onlyfigures.'

2.1.2 breast_cancer_ishodnik Binarization

- The 'mean_radius' feature was divided into bins starting from 6.981 - 9.981 to 27.981 - 30.981 with 3-unit increments, followed by one-hot encoding.
- 'Mean_texture' was divided into bins starting from 10.38 - 15.38 to 30.38 - 35.38 with 5-unit increments, followed by one-hot encoding.
- 'Mean_perimeter' was divided into bins starting from 43.79 - 63.79 to 183.79 - 203.79 with 20-unit increments, followed by one-hot encoding.
- 'Mean_smoothness' was divided into bins from 0.05 - 0.07 to 0.015 - 0.17 with 0.02 increments, followed by one-hot encoding.

The resulting dataset, 'breast_cancer_binarized,' consisted of 500 rows and 40 columns.

2.1.3 heart_failure Binarization

- The 'age' feature was binarized using bins [0, 20, 40, 60, 80] and one-hot encoding.
- 'Creatine_phosphokinase' underwent one-hot encoding with bins [0, 120, 250].
- 'Ejection_fraction' was divided into bins [0, 40, 55].
- Similar binarization was applied to 'platelets' with bins [0, 150000, 450000].
- 'Serum_creatinine' underwent binarization with bins [0, 0.7, 1.2].
- 'Sodium' was binarized using bins [0, 135, 145].

The final dataset, 'heart_stroke_binarized,' consisted of 500 rows and 29 columns.

3 ML methods application

Binary classification was performed on the datasets: ‘heart_stroke_onlyfigures‘, ‘heart_stroke_binarized‘, ‘breast_cancer_ishodnik‘, ‘breast_cancer_binarized‘, ‘heart_failure_ishodnik‘, and ‘heart_failure_binar
This section describes the specific features and details of classifications by each method on these datasets. The implementation code is available at https://github.com/dealvost/OSDA-Big-HW.

3.1 Binarized Binary Classification (alpha-weak)

Applied to the fully binarized datasets: ‘heart_stroke_binarized‘, ‘breast_cancer_binarized‘, and ‘heart_failure_binarized‘.

3.2 Binarized Binary Classification (alpha-weak-support)

Also used on the fully binarized datasets.

3.3 Binarized Binary Classification (ratio-support)

Applied to the same fully binarized datasets.

3.4 Pattern Binary Classification (alpha-weak)

Used on ‘heart_stroke_onlyfigures‘, ‘breast_cancer_ishodnik‘, and ‘heart_failure_ishodnik‘, with prior specification of categorical variable columns.

3.5 Pattern Binary Classification (alpha-weak-support) and (ratio-support)

Both methods were applied to ‘heart_stroke_onlyfigures‘,
‘breast_cancer_ishodnik‘, and ‘heart_failure_ishodnik‘, with categorical variables specified beforehand.

3.6 Decision Tree

Applied to ‘heart_stroke_onlyfigures‘, ‘breast_cancer_ishodnik‘, and ‘heart_failure_ishodnik‘. Hyperparameters were optimized using GridSearchCV. The best hyperparameters for each dataset were:

- ‘heart_stroke_onlyfigures‘: ’criterion’: ’gini’, ’max_depth’: 4, ’min_samples_leaf’: 6, ’min_samples_split’: 2.
- ‘breast_cancer_ishodnik‘: ’criterion’: ’gini’, ’max_depth’: 6, ’min_samples_leaf’: 5, ’min_samples_split’: 2.
- ‘heart_failure_ishodnik‘: ’criterion’: ’gini’, ’max_depth’: 1, ’min_samples_leaf’: 1, ’min_samples_split’: 2.

3.7 Random Forest

Used on the same datasets, with hyperparameters optimized using GridSearchCV. Optimal settings for each dataset were:

- 'heart_stroke_onlyfigures': 'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 10.
- 'breast_cancer_ishodnik': 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50.
- 'heart_failure_ishodnik': 'max_depth': 10, 'min_samples_leaf': 8, 'min_samples_split': 5, 'n_estimators': 10.

3.8 XBoost

Implemented on 'heart_stroke_onlyfigures', 'breast_cancer_ishodnik', and 'heart_failure_ishodnik', with hyperparameter tuning through GridSearchCV, yielding the following best settings:

- 'heart_stroke_onlyfigures': 'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 10.
- 'breast_cancer_ishodnik': 'colsample_bytree': 0.8, 'gamma': 0.5, 'learning_rate': 0.3, 'max_depth': 3, 'n_estimators': 50, 'subsample': 1.
- 'heart_failure_ishodnik': 'colsample_bytree': 0.8, 'gamma': 0, 'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 50, 'subsample': 0.8.

3.9 Catboost

Implemented on 'heart_stroke_onlyfigures', 'breast_cancer_ishodnik', and 'heart_failure_ishodnik' without prior hyperparameter tuning.

3.10 k-NN

Applied to the same datasets, using StandardScaler to mitigate the impact of data scaling issues. Hyperparameters were optimized using GridSearchCV, resulting in:

- 'heart_stroke_onlyfigures': 'knn__metric': 'euclidean', 'knn__n_neighbors': 2.
- 'breast_cancer_ishodnik': 'knn__metric': 'manhattan', 'knn__n_neighbors': 14.
- 'heart_failure_ishodnik': 'knn__metric': 'euclidean', 'knn__n_neighbors': 7.

3.11 Bernoulli Naive Bayes

Employed on the fully binarized datasets with the best hyperparameters:

- 'heart_stroke_binarized': 'alpha': 30.
- 'breast_cancer_binarized': 'alpha': 0.01.
- 'heart_failure_binarized': 'alpha': 10.

3.12 Logistic Regression

Used on the same datasets, with StandardScaler to address data scale issues. Hyperparameter tuning was done using GridSearchCV, resulting in:

- 'heart_stroke_onlyfigures': 'logreg__C': 0.001, 'logreg__penalty': 'l2'.
- 'breast_cancer_ishodnik': 'logreg__C': 100, 'logreg__penalty': 'l2'.
- 'heart_failure_ishodnik': 'logreg__C': 1, 'logreg__penalty': 'l2'.

4 Cross-Validation and Visualization of Results

The KFold method is utilized for splitting the data into 5 parts (`n_splits=5`), facilitating a 5-fold cross-validation process. Shuffling (`shuffle=True`) is enabled with a fixed `random_state` to ensure reproducibility of results.

4.1 Training Sample Size Variation Cycle

The sizes of the training samples vary from 10% to 100% of the original data. For each sample size:

- Cross-validation is conducted: data is split into training and test sets.
- A fraction of the training data, corresponding to the current `frac` value, is utilized.
- A decision tree model is trained on this subset.
- Predictions are made on the test set, and the time taken for prediction is measured.

4.2 Aggregation and Averaging of Results

- For each cross-validation iteration, accuracy and F1-score of predictions are calculated.
- The time spent on predictions is recorded.
- Results (accuracy, F1-score, prediction time) are averaged across all iterations of cross-validation for each sample size.

4.3 Graph Construction

Upon completing the cross-validation, three graphs are constructed:

4.3.1 Accuracy Score Progression Graph

- X-Axis: Training sample sizes (`dt_train_sizes`).
- Y-Axis: Averaged accuracy scores (`dt_accuracy_scores`).
- Illustrates how the model's accuracy varies with the size of the training sample.

4.3.2 F1 Score Progression Graph

- X-Axis: Training sample sizes.
- Y-Axis: Averaged F1 scores (`dt_f1_scores`).
- Shows the variation in F1 score relative to the size of the training sample.

4.3.3 Prediction Time Progression Graph

- X-Axis: Training sample sizes.
- Y-Axis: Averaged prediction times (`dt_prediction_times`).
- Demonstrates how the time required for making predictions changes depending on the training sample size.

These graphs enable the evaluation of the model's quality (through accuracy and F1 score) and performance (through prediction time) based on the volume of data used for training.

5 Obtained Results and Analysis

5.1 Heart Failure Dataset

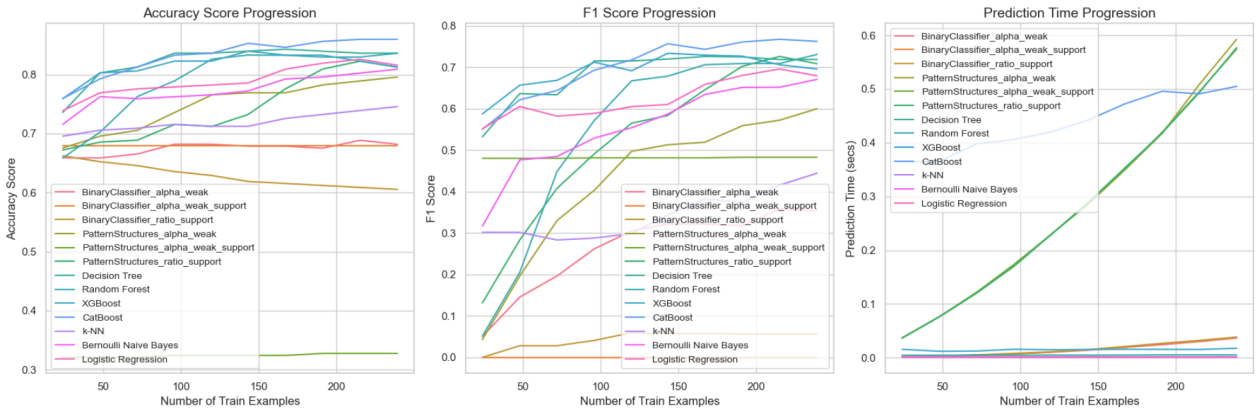

Figure 1: Comparison of Machine Learning Model Performances for the Heart Failure Dataset

It can be observed that the CatBoost model demonstrates the highest effectiveness in terms of both the F1 and accuracy metrics. The Pattern Structures_ratio_support model ranks fifth, following the CatBoost, Decision Tree, Random Forest, Logistic Regression, and XGBoost models in terms of accuracy, and it holds the fourth position in the F1 score metric, coming after the CatBoost, Random Forest, and Decision Tree models.

The other models, particularly those in the BinaryClassifier and PatternStructures categories, occupy the lower echelons in these metrics and are unable to compete with the classical machine learning methods.

From a time consumption perspective, the PatternStructures_ratio_support approach is one of the most time-intensive, ranking second-to-last in this criterion. The CatBoost method stands at the 11th position in the list of most time-efficient methods.

Despite the Pattern Structures_ratio_support model providing the best competition to classical machine learning methods on this dataset, none of the algorithms explored from the FCALC library demonstrate superiority over existing approaches in machine learning, based on the obtained data.

5.2 Breast Cancer Dataset

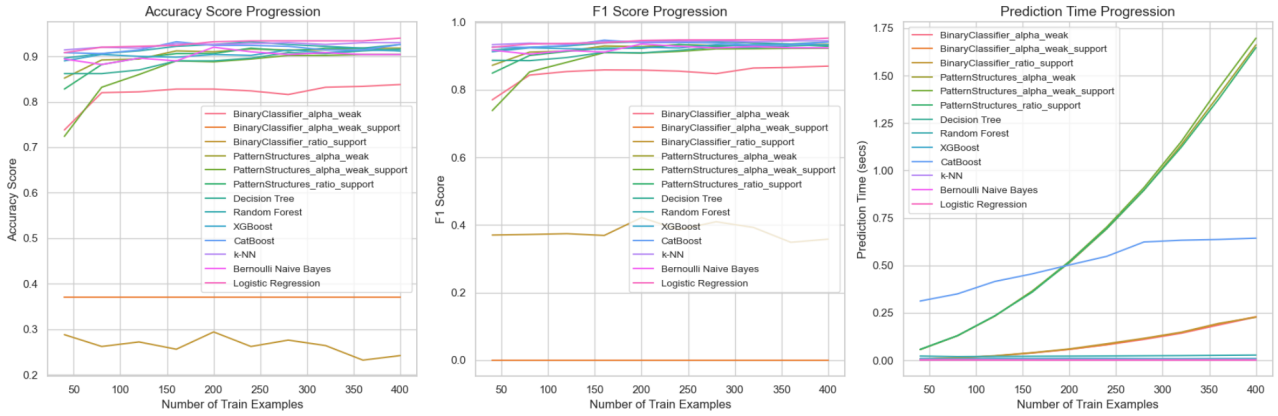

Figure 2: Comparison of Machine Learning Model Performances for the Breast Cancer Dataset

Logistic Regression ranks first in terms of both accuracy and F1-score.

The PatternStructures_alpha_weak method is positioned fourth in accuracy following the k-NN, CatBoost, and RandomForest methods, and eighth in the F1-score metric. The PatternStructures_ratio_support method stands eighth in both accuracy and F1-score.

PatternStructures_alpha_weak secures the fourth position in F1-score after Logistic Regression, k-NN, and CatBoost, while it almost occupies the penultimate place in accuracy.

In terms of prediction time, Logistic Regression proves to be one of the most efficient, securing the third position, whereas all FCA methods fall into the lower half of the ranking.

It can be concluded that none of the methods from the FCALC library can compete with the traditional machine learning methods on this dataset.

5.3 Heart Stroke Dataset

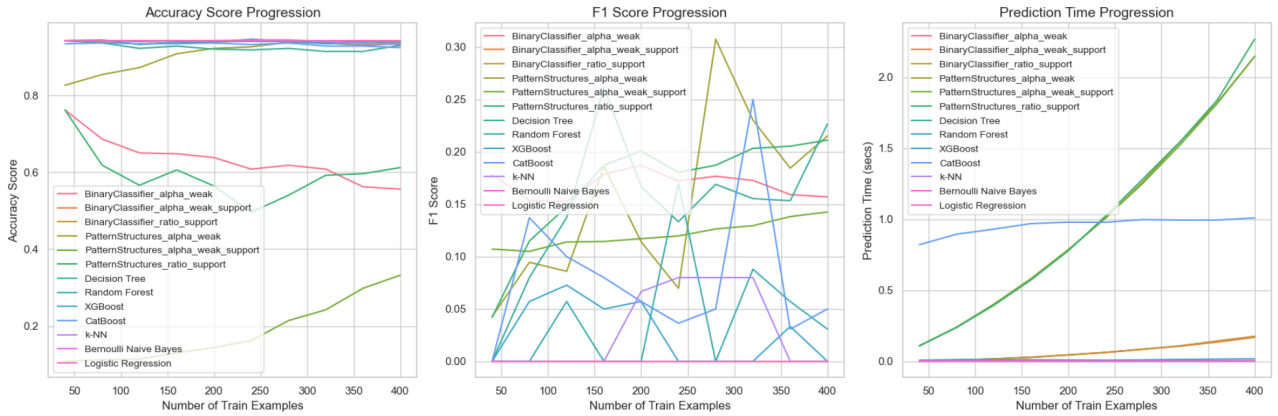

Figure 3: Comparison of Machine Learning Model Performances for the Heart Stroke Dataset

The "peculiar"behavior of the curves on the graphs is attributed to the imbalance in this dataset: it contains only 29 ones and 471 zeros in the target variable values.

The BinaryClassifier_alpha_weak_support method ranked first in accuracy and fifth in F1-score.

The BinaryClassifier_ratio_support method occupies the fourth position in accuracy and the last in F1-score.

The PatternStructures_alpha_weak method is sixth in accuracy (lagging behind the leader by only 0.007 in absolute terms) and second in the F1-score ranking.

All methods from the FCALC library are in the lower half of the ranking in terms of Prediction time.

Based on the results, it can be said that on this dataset, some FCALC methods (with PatternStructures_alpha_weak showing the best performance) provide substantial competition to traditional ML methods and outperform them in several aspects.

6 Conclusion

Within the scope of this study, methods (some of the methods) from the FCALC library demonstrated the best results on the Heart Stroke Dataset, which was the most imbalanced among all examined datasets.

However, for the other two datasets, traditional ML methods showed superior results, leaving no room for competition from FCALC methods.

It seems prudent to further investigate the efficacy of FCALC library methods on imbalanced datasets. According to the results obtained, these methods could potentially offer competition to traditional ML methods on such types of datasets.