SUMMER INTERNSHIP

PROJECT REPORT 2022

comparing several libraries for constructing auto-neural networks

Internship Details:

Internship Start Date: 24th May, 2022

Internship End Date: 7th Aug,2022

Mentor:

Prof. Amit Mankodi

DA-IICT, Gandhinagar

Gujarat-382007

Submitted By:

Jadhav Shailesh	202112051
Radadiya Jay	202112121

Table of Contents

1. Introduction **3**
1.1 Introduction 3
1.2 Objective 3
1.3 Scope 3
1.4 Definition, Acronyms, and Abbreviation 3
1.5 Technologies 4

2. Preliminary **5**
2.1 RMSE 5
2.2 MedAE 5
2.3 MedPAE 5

3. Implementation **6**
3.1 Auto-Sklearn 6
3.1.1 Result analysis of Auto-Sklearn 7
3.2 Auto-PyTorch 8
3.2.1 Result analysis of Auto-PyTorch 8
3.3 TPOT 10
3.3.1 Result analysis of TPOT 10
3.4 MLJAR 12
3.4.1 Result analysis of MLJAR 12

4. Conclusion **14**

1. Introduction:

1.1 Introduction:

A Machine Learning (ML) application includes typically several steps: data preparation, feature engineering, algorithm selection and hyperparameter tuning. Most of these steps require trial and error approaches, in the last years there has been an attempt to automate several components of the ML workflow, giving rise to the concepts of Automated Machine Learning (AutoML) This report focuses on the comparisions of the best supervised AutoML Libraries

1.2 System Objective:

Four contemporary open-source AutoML technologies are taken into account in the comparative study with our customized AutoML model. We utilize various datasets to assess these libraries.

1.3 System Scope:

Auto-PyTorch, Auto-Sklearn, TPOT, MLJAR are the four best libraries we have compared with our AutoML model. We have dataset with details on various system specs. Dataset contains details on things like CPU clock speed, cache memory size, memory type, and more. In order to make the data more usable for future study, we deleted several aspects that weren't essential for our trials.

1.4 Definitions, Acronyms and Abbreviations:

- **RMSE**: - Root Mean Square Error
- **MedAE**: - Median Absolute Error
- **MedAPE**: - mean absolute percent erro

1.5 Tools And Technologies:

- Google Colab
- Python 3.7

1.5.1. Libraries:

- Pandas
- Category_encoders
- Sklearn
- Numpy
- TPOT
- MLJAR
- PyTorch

2. Preliminary:

For comparison we have calculated the error count for this model with our dataset.

2.1. RMSE: Root Mean Square Error (RMSE) is the residuals' standard deviation (prediction errors). The distance between the data points and the regression line is measured by residuals, and the spread of these residuals is measured by RMSE. In other words, it provides information on how tightly the data is clustered around the line of best fit. To validate the outcomes of experiments, root mean square error is frequently utilized in climatology, forecasting, and regression analysis.

$$\boldsymbol{RMSC = \sqrt{\frac{\sum_{i=1}^{N}(predicted_i - Actual_i)^2}{N}}}$$

2.2. **MedAE:** The median absolute error is particularly interesting because it is robust to outliers. The loss is calculated by taking the median of all absolute differences between the target and the prediction. The Median Absolute Deviation (MAD) is a simple way to quantify variation. Half the values are closer to the median than the MAD, and half are further away

$$MedAE = \frac{\sum_{i=1}^{N} |predicted_i - Actual_i|}{N}$$

2.3. **MedAPE:** The mean absolute percent error (MEDAPE) expresses accuracy as a percentage of the error. Because the MAPE is a percentage, it can be easier to understand than the other accuracy measure statistics.

$$MEDAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{(predicted_i - Actual_i)}{predicted_i}\right|$$

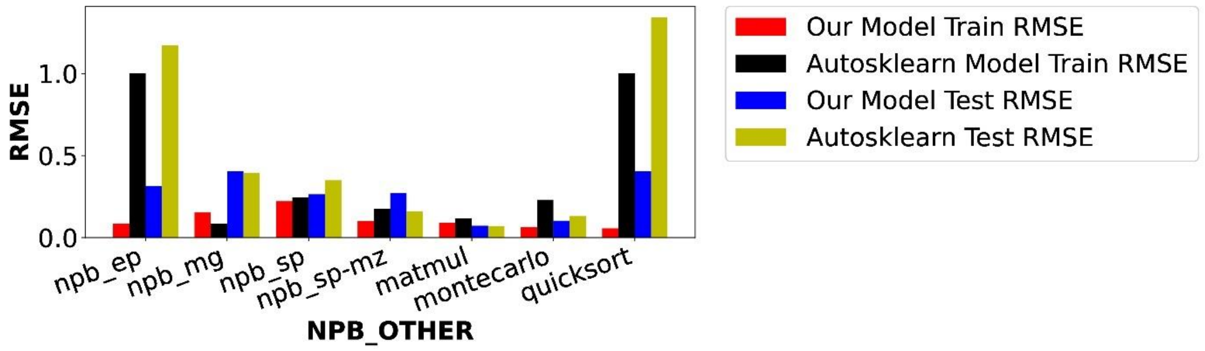
3. Implementation:

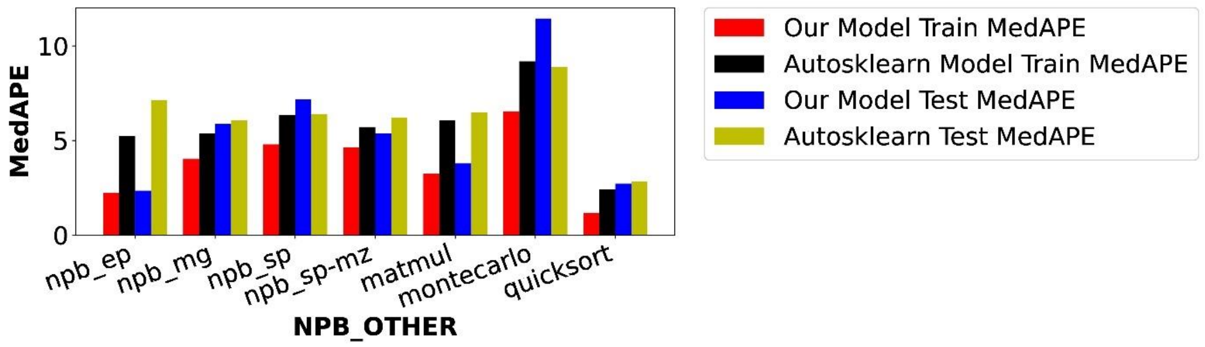
3.1 Auto-Sklearn:

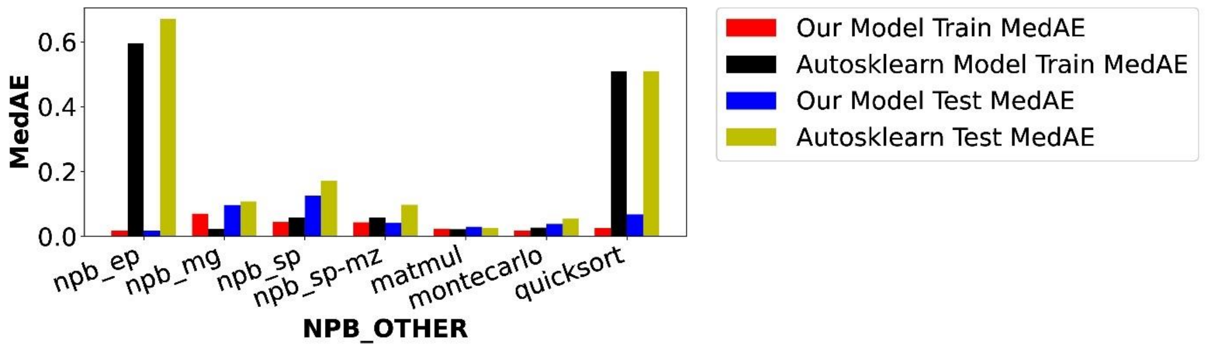
- By automatically accounting for prior performance on comparable datasets and creating ensembles from the models considered during the optimization, AUTO-SKLEARN outperforms conventional AutoML approaches
- You only need to construct and setup an instance of the AutoSklearnRegressor class, fit it to your dataset, and that's all whether your prediction goal is classification or regression. Once created, the model may either be utilised immediately to generate predictions or saved to a file (using pickle) for use at a later time.
- The AutoSklearn class accepts several configuration settings as inputs.
- A train-test split of your dataset will be used by default during the search, and this default is advised for both speed and simplicity.
- The optimization procedure will continue for the number of minutes that you specify. It will run for an hour by default.
- Setting the "time left for this task" parameter to the duration you want the procedure to take is advised.
- After implementing this AutoSklearn we compared this result with our neural network code and generated result comparison graph of RMSE, MEDAE,MEDPAE
- CODE LINK

3.1.1. Result Analysis of Auto-Sklearn

- Here, it is quite evident that our model test outperforms the auto-sklearn.Only in npg_mg,matmul have little differ results in medAE.
- AutoSklearn modelling estimates that it will take 5 to 10 minutes to create a neural network, our model can be created in as little as 20 to 24 seconds.

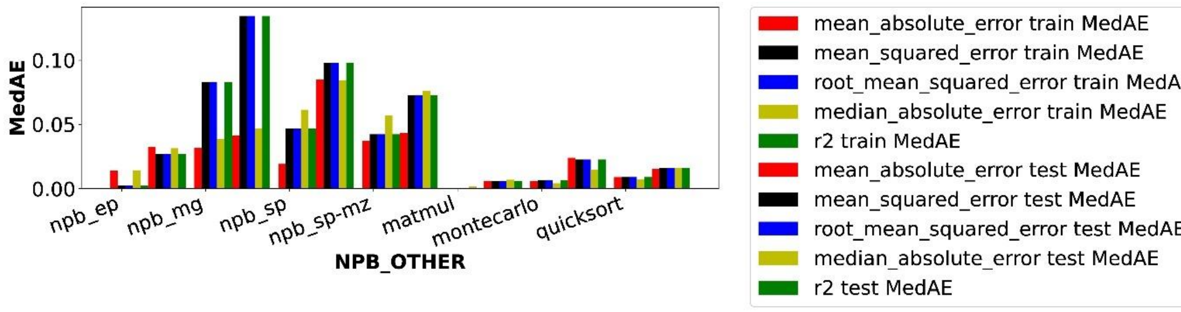






3.2 Auto-PyTorch

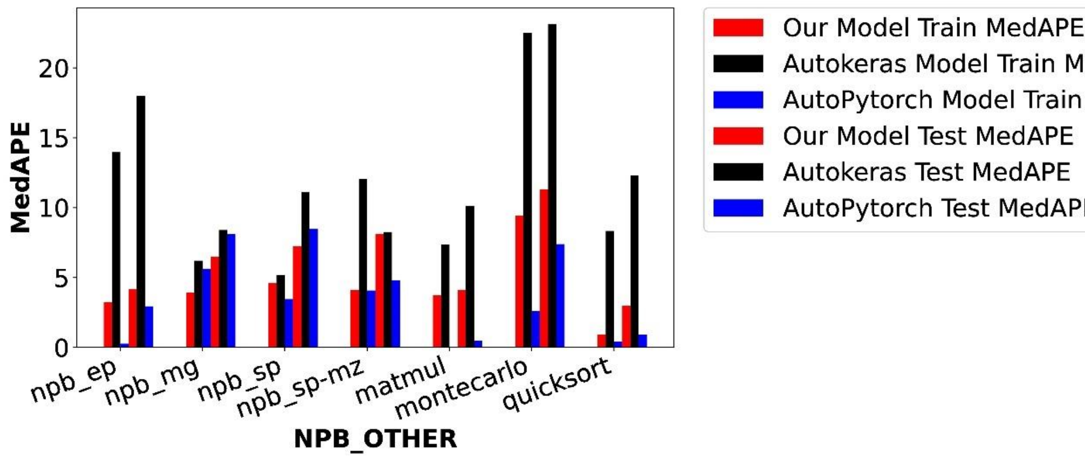
- Auto-PyTorch was primarily created to support time series data and tabular data (classification, regression) (forecasting). The paper "Auto-PyTorch Tabular: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL" describes the newest features of Auto-PyTorch for tabular data
- It uses several techniques to create neural networks , comparison of all techniques' rmse, medae,medaep

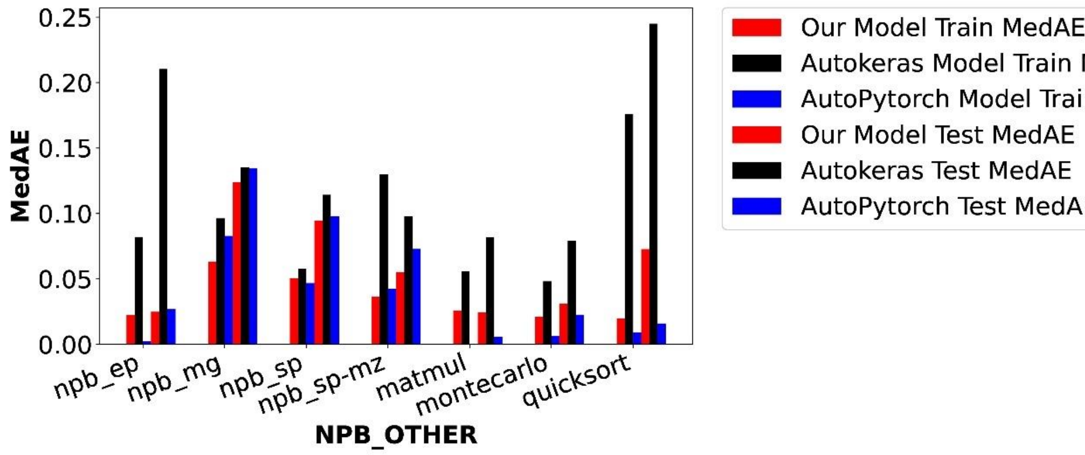


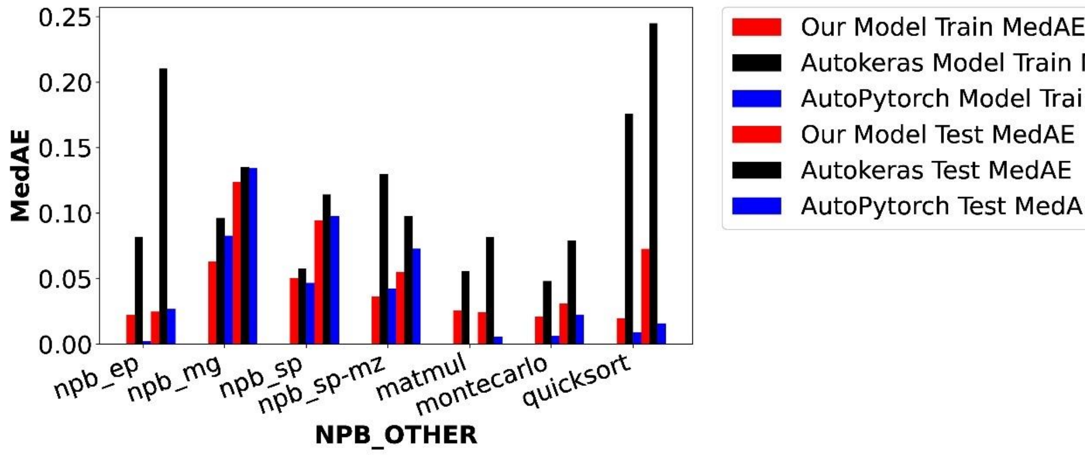
-
- You can see it by clicking here
- And we selected the finest of them to compare with our model, and once more for this purpose, we produced the graph.
- Code Link

3.2.1. Result Analysis of Auto-Pytorch

- In comparison to autokeras, autopytorch has demonstrated higher results for its best setup, but it is unable to outperform our model overall. This is true even though it took auto-pytorch around 6 minutes to develop and test compared to our model's approximate 20 seconds

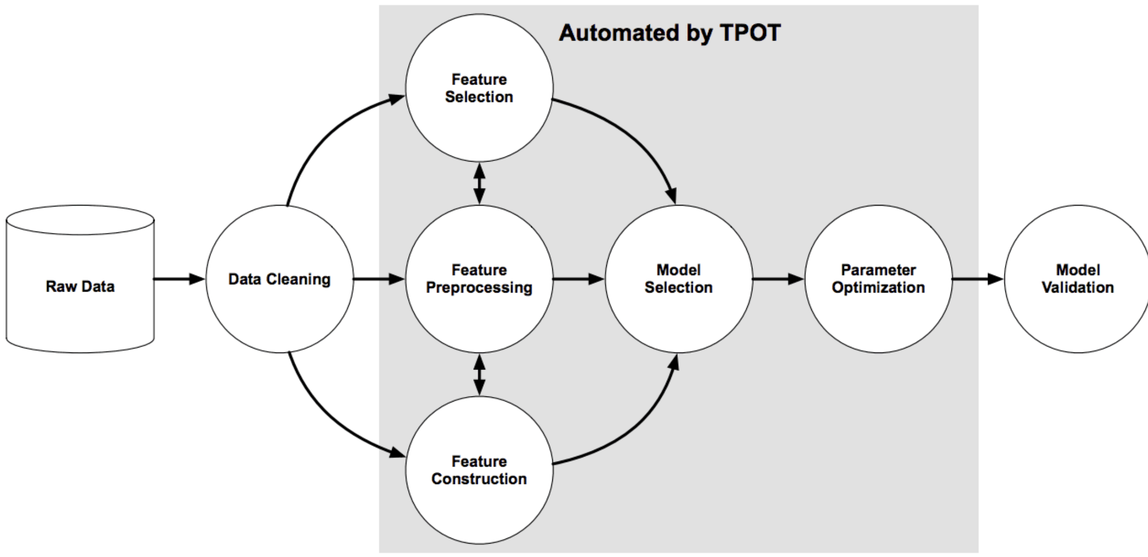
- 
MedAPE
NPB_OTHER
npb_ep
npb_mg
npb_sp
npb_sp-mz
matmul
montecarlo
quicksort
Our Model Train MedAPE
Autokeras Model Train M
AutoPytorch Model Train
Our Model Test MedAPE
Autokeras Test MedAPE
AutoPytorch Test MedAPI

- 
MedAE
NPB_OTHER
npb_ep
npb_mg
npb_sp
npb_sp-mz
matmul
montecarlo
quicksort
Our Model Train MedAE
Autokeras Model Train I
AutoPytorch Model Trai
Our Model Test MedAE
Autokeras Test MedAE
AutoPytorch Test MedA

- 
MedAE
NPB_OTHER
npb_ep
npb_mg
npb_sp
npb_sp-mz
matmul
montecarlo
quicksort
Our Model Train MedAE
Autokeras Model Train I
AutoPytorch Model Trai
Our Model Test MedAE
Autokeras Test MedAE
AutoPytorch Test MedA

3.3 TPOT

- In an effort to increase classification accuracy for a certain piece of supervised learning data, TPOT automatically designs and optimises a number of data transformations and machine learning models.
- The aspects of the pipeline search, such as data cleaning, feature selection, feature processing, feature creation, model selection, and hyperparameter optimization, are shown in the image below, which is taken from the TPOT article.

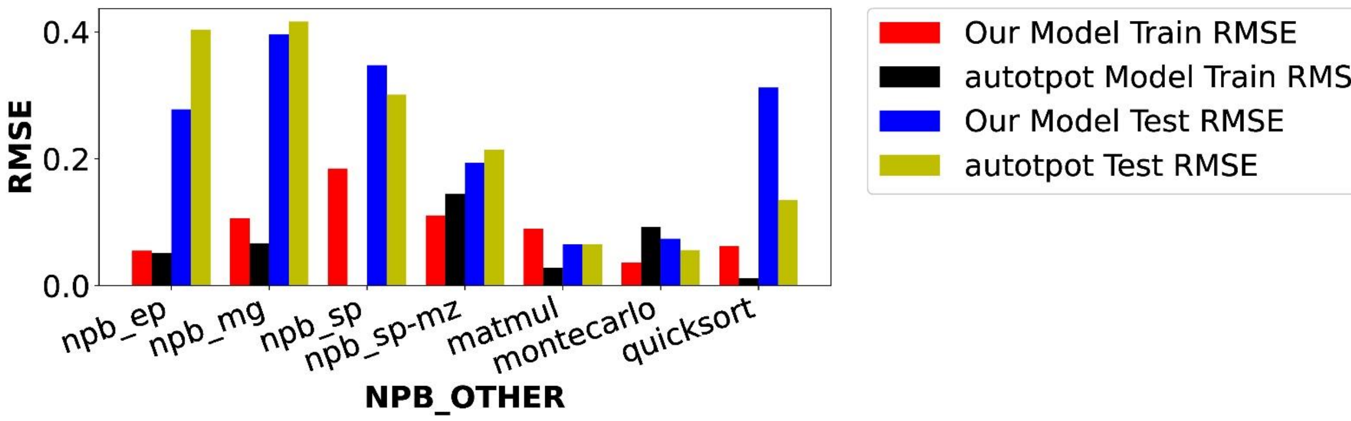


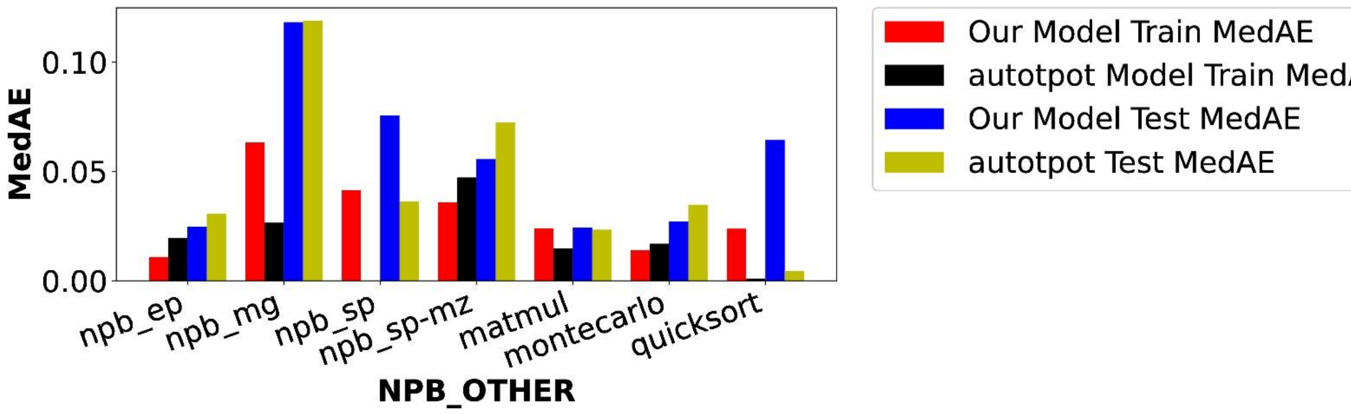
-
- For the search, we'll use a population size of 50 for 5 generations, and by setting "n jobs" to -1, we'll employ all of the system's cores.
- Code Link

3.3.1. Result Analysis of TPOT

- The time required to execute the Tpot model is more than 10-15 minutes for each dataset, which is significantly longer than our estimate. However, Tpot has demonstrated decent results overall, with the exception of a few datasets (our model take

20 sec max for each dataset).



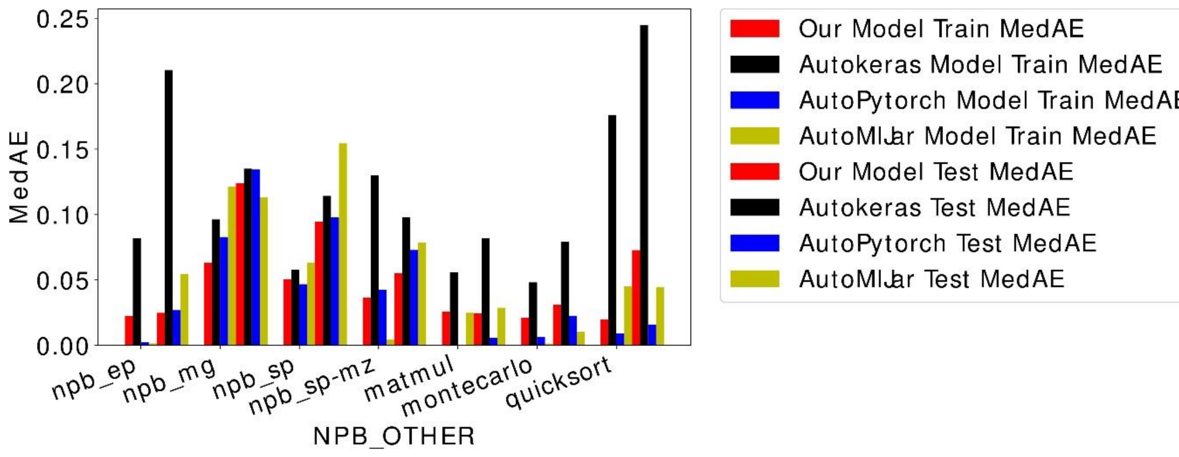


3.4 MLJAR

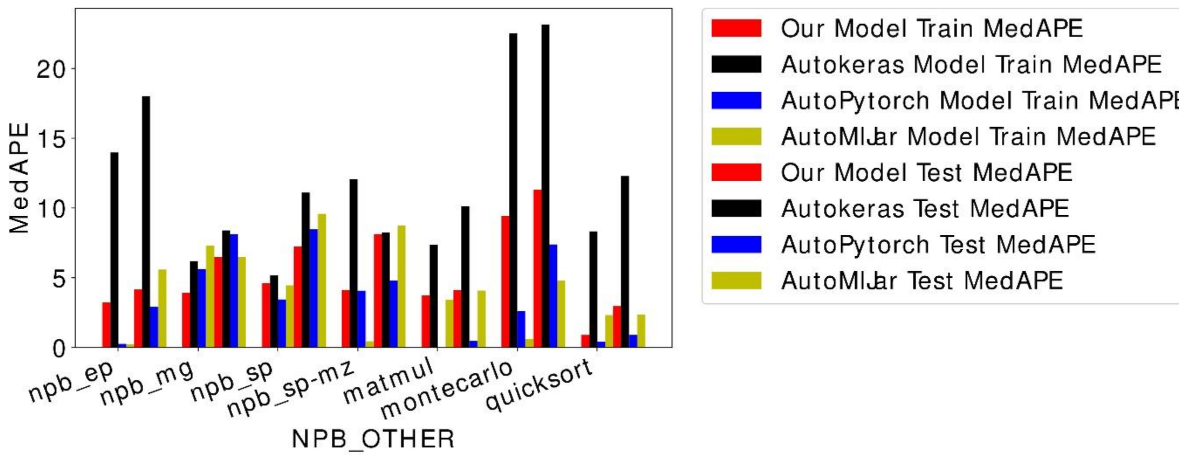
- A Python library for Automated Machine Learning that uses tabular data is called mljar-supervised. It encapsulates the typical methods for building machine learning models, preprocessing data, and modifying hyper-parameters to get the optimal model. You can see exactly how the ML pipeline is built, thus it is not a "black box."
- Code Link

3.4.1. Result Analysis of MLJAR

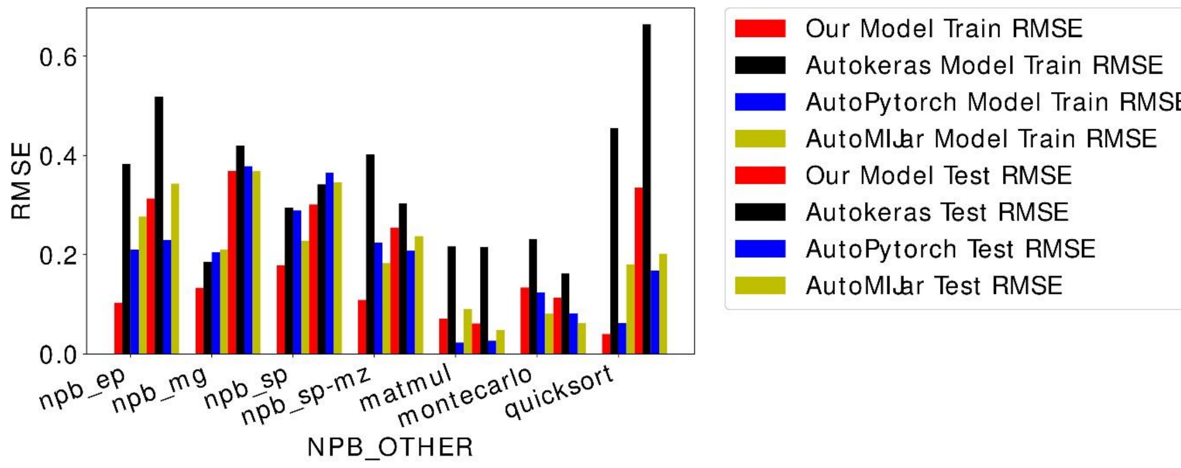
- In comparison to autokeras, autopytorch and MLJAR . MLJAR has demonstrated higher results for its best setup, but it is unable to outperform our model overall.



-



-


RMSE
0.0
0.2
0.4
0.6
npb_ep
npb_mg
npb_sp
npb_sp-mz
matmul
montecarlo
quicksort
NPB_OTHER
Our Model Train RMSE
Autokeras Model Train RMSE
AutoPytorch Model Train RMS
AutoMlJar Model Train RMSE
Our Model Test RMSE
Autokeras Test RMSE
AutoPytorch Test RMSE
AutoMlJar Test RMSE

➢

4. Conclusion:

- After implementing every model in its best configuration and contrasting them all with our model, we find that, with the exception of a few datasets, our model performed well. However, the tpot model performed slightly better than our model, and since tpot's network-building time was significantly longer than our model's for these datasets, we must conclude that, on the whole, our model is faster and more effective.