All Tools On Deck

Supriya, Attila, Isak
Use_your_imagination

Modality 1/2

- 🟩 Week 1
- ☐ Week 2
- ☐ Week 3
- 🟩 Week 4
- ☐ Week 5
- ☐ Week 6
- ☐ Week 7
- ☐ Week 8
- 🟩 Week 9
- 🟩 Week 10
- 🟩 Bonus
- 🟩 Literature

Motivation

AutoGluon is a cutting-edge tool for automating machine learning (AutoML) processes on tabular datasets. The objective is to investigate the performance of several AutoML tools, including auto-sklearn, FLAML, TPOT and MLJAR in comparison to AutoGluon's predictive capabilities.

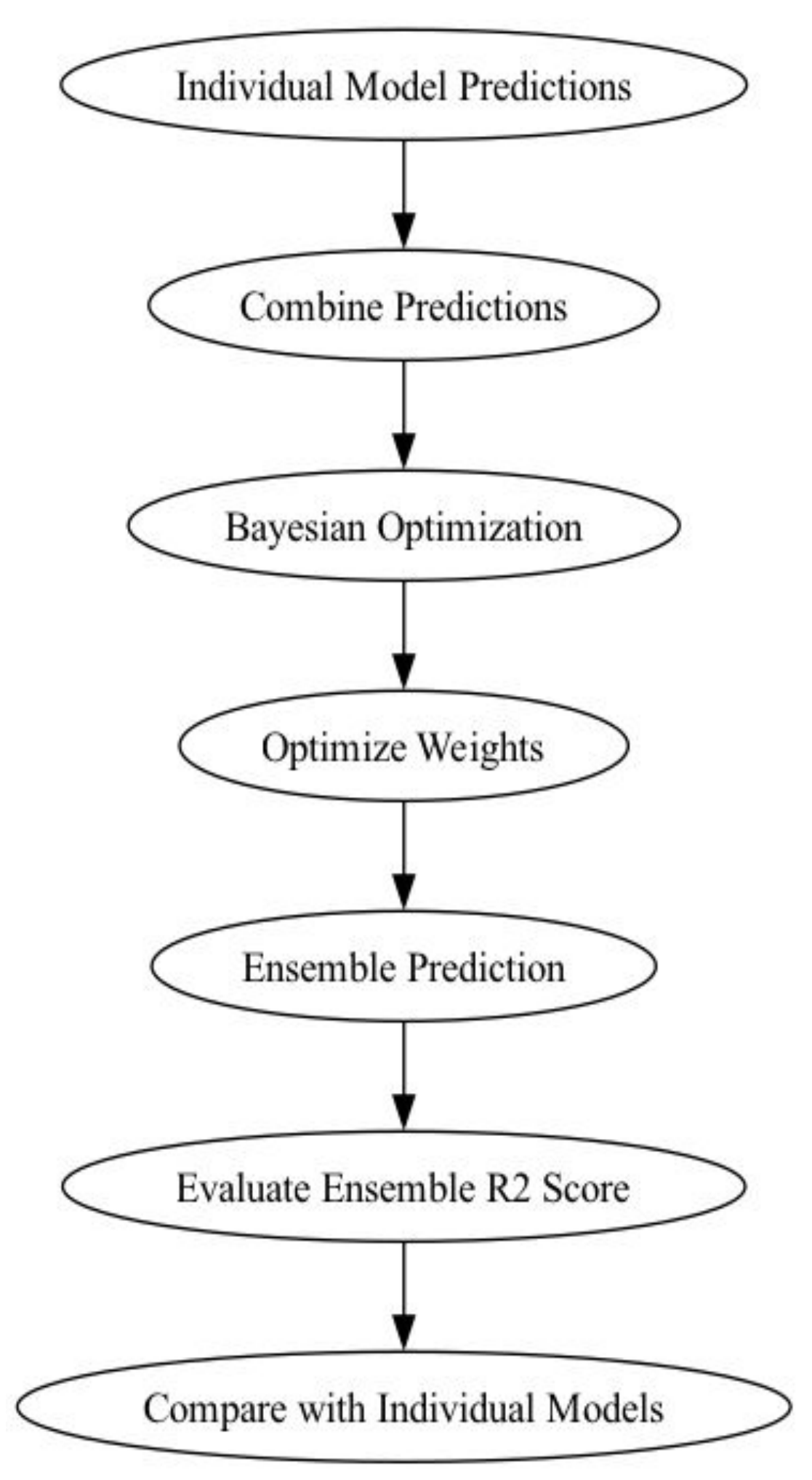
While AutoGluon typically excels in longer training durations, our focus lies in examining its performance within shorter training times, specifically less than a minute. We aim to contrast this with an AutoGluon predictor trained for an equivalent total duration.

Experimental Configuration

Configurations

1. **Random Seeds**:
   - Experiment with seed 42 to ensure the robustness of results.
2. **Time Limits**:
   - Use time limits of 45 seconds to evaluate the impact of computation time on model performance.
3. **Frameworks and Tools**:
   - **TPOT**: Genetic programming for automated feature selection and model optimization.
   - **MLJAR**: User-friendly interface with automated machine learning pipelines.
   - **AutoGluon**: Robust framework with high performance on tabular data.
   - **FLAML**: Efficient automated machine learning library focusing on low computational cost.
   - **AutoSKLearn**: Ensemble learning with efficient hyperparameter optimization.

Ensemble Optimization Process


Individual Model Predictions
Combine Predictions
Bayesian Optimization
Optimize Weights
Ensemble Prediction
Evaluate Ensemble R2 Score
Compare with Individual Models

Timeline

- Baseline: **Simple averaging** with AutoGluon and AutoSKLearn
  - Suboptimal to a pure Gluon prediction
- Sequential **tool integration** of FLAML, MLJAR and TPOT
  - Seeing sharp improvement on ensemble with FLAML and MLJAR for specific sets
- **Sophisticating** ensemble method (normalized initial weights

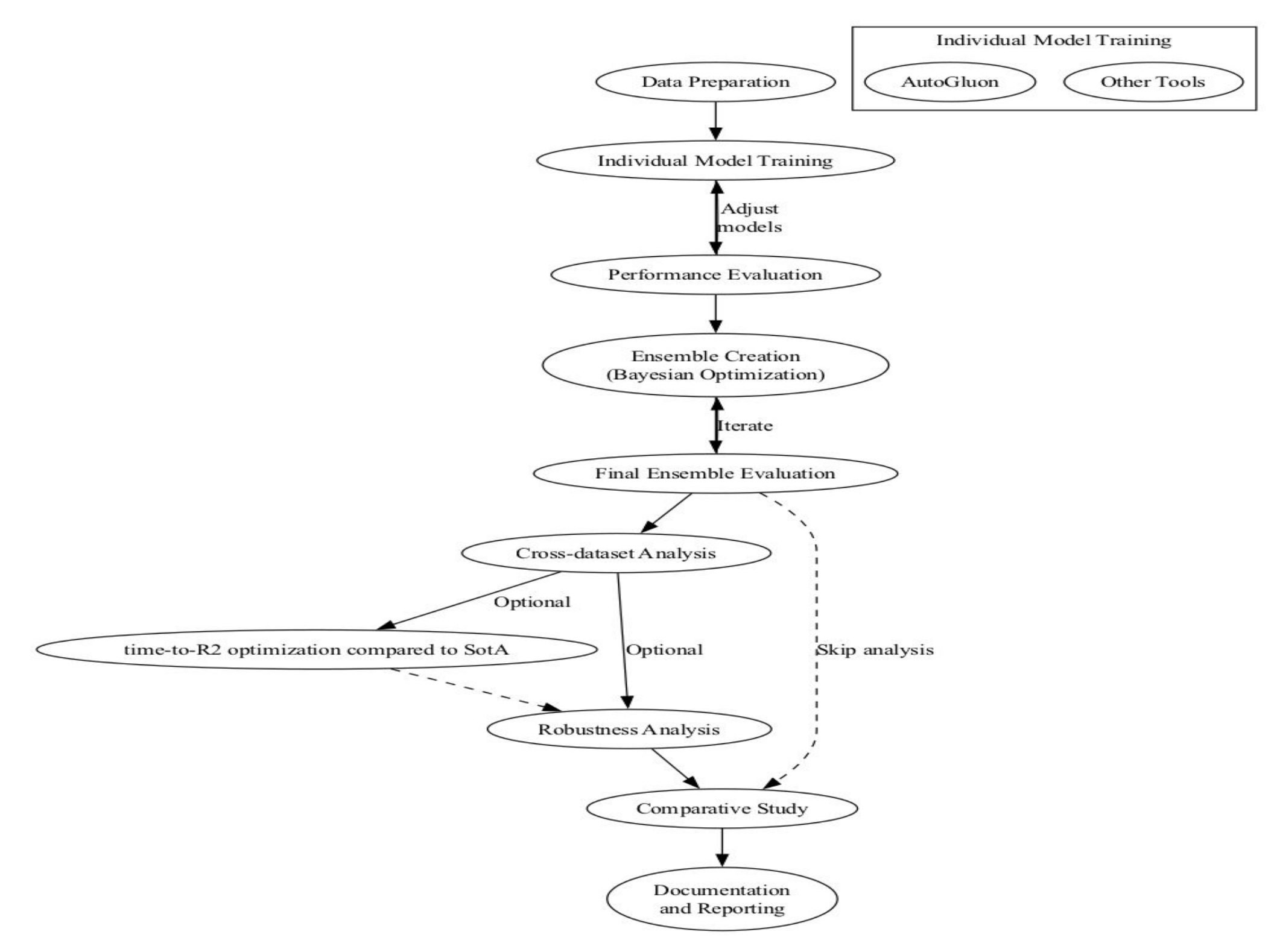
```
# Define the search space
space = [Real(max(0, w-0.3), min(1, w+0.3), name=f'w{i+1}') for i, w in enumerate(initial_weights)]
# Set up the optimization
res = gp_minimize(objective, space, n_calls=50, random_state=random_seed, x0=initial_weights)
```

  - Followed by summating, normalizing, applying
- **Removing TPOT** given experiment results
- Using 0.8/0.2 train-test split to figure out **"optimal" weights** for final exam predictions

In total a series of 21 ensemble experiments training tools on different time, sample-seeds and datasets

Approach

Our approach involves aggregating the various AutoML tools into a predictor ensemble using Bayesian optimization. This will enable us to find the optimal weights for each predictor, taking into account the significant variability in performance across different datasets.


Individual Model Training
AutoGluon
Other Tools
Data Preparation
Individual Model Training
Adjust models
Performance Evaluation
Ensemble Creation (Bayesian Optimization)
Iterate
Final Ensemble Evaluation
Cross-dataset Analysis
Optional
time-to-R2 optimization compared to SotA
Optional
Skip analysis
Robustness Analysis
Comparative Study
Documentation and Reporting

Results

Example experiment result

Experiment 14 (exam set), split train/test: 0.2, random_seed=42, 45 sec each

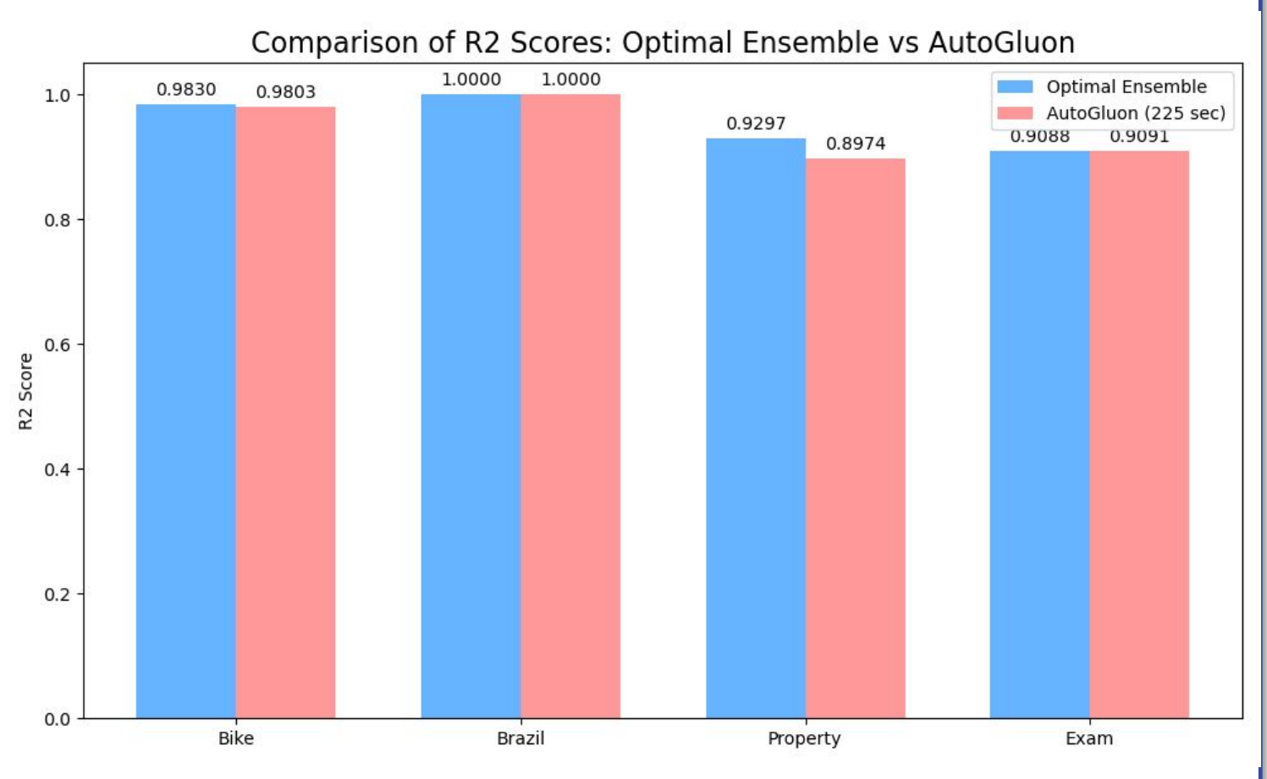
autosklearn r2: 0.90189, flaml r2: 0.90217, gluon r2: 0.90824, tpot r2: 0.85345, mljar r2: 0.90662

Initial weights: FLAML: 0.202, AutoSklearn: 0.202, Gluon: 0.203, TPOT: 0.2031, MLJAR: 0.20308

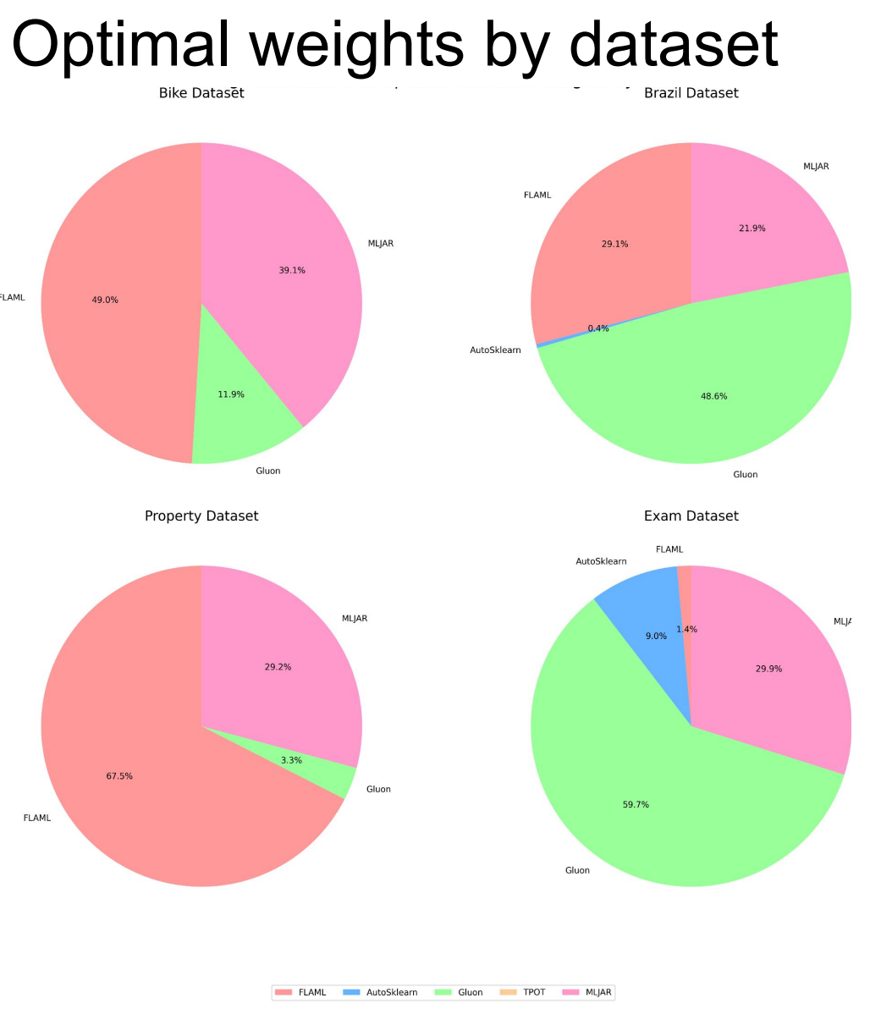
Optimal weights: FLAML: 0.000, AutoSklearn: 0.067, Gluon: 0.584, TPOT: 0.0000, MLJAR: 0.34891

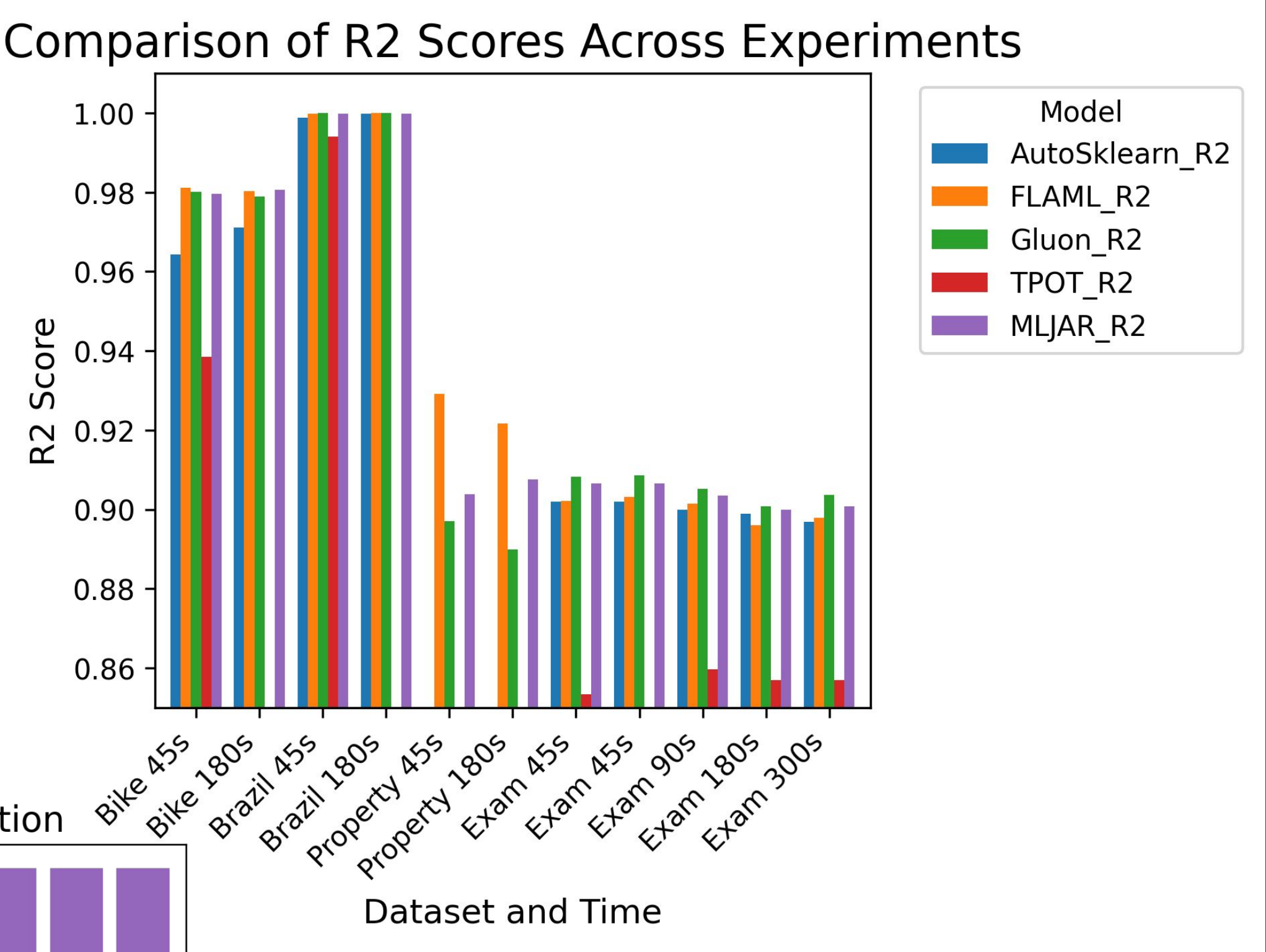
Optimal ensemble R2: 0.908760

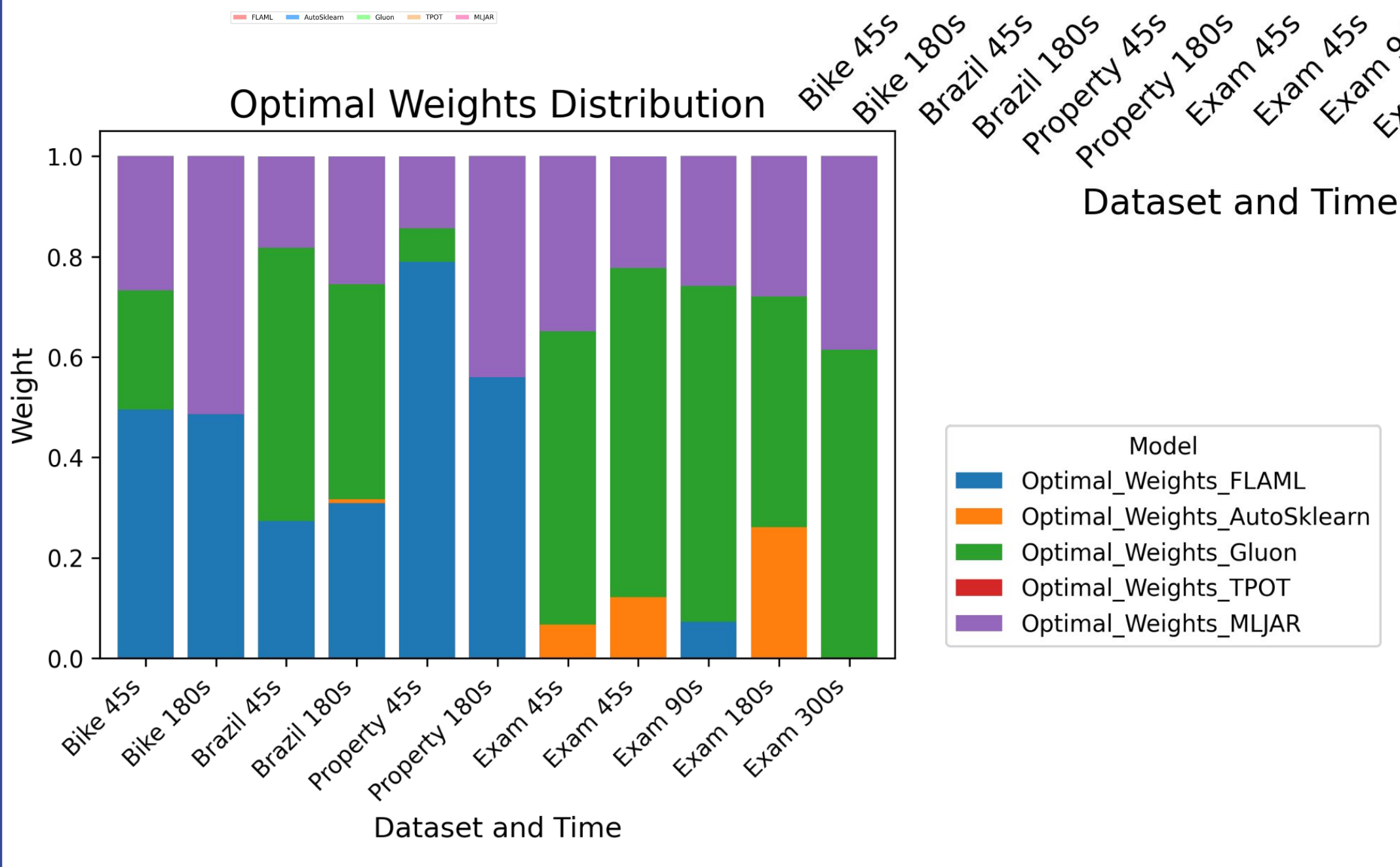
Compared with a 225 sec autogluon run on same dataset. 0.909096

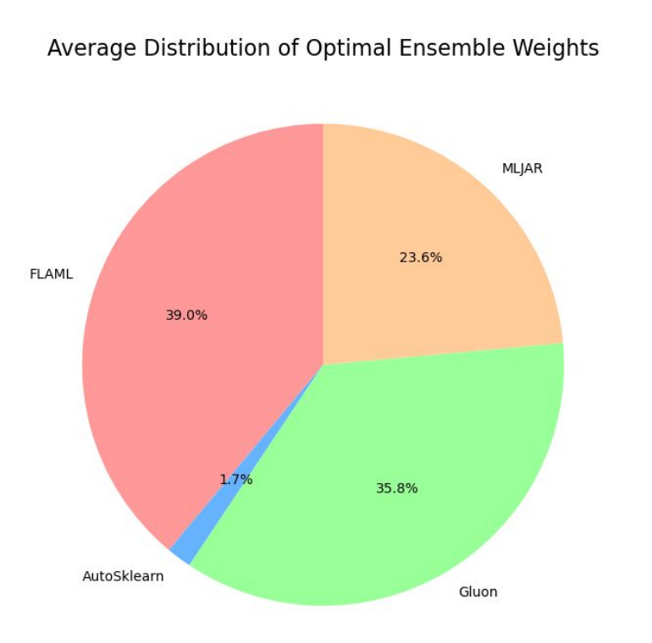


Optimal weights by dataset









Resources Used

For development:
- Google Colab
- Sage Maker Studio Lab
- Total compute estimate: 100 CPU-h

For AutoML:
- Google Colab
- SageMaker
- 225 sec

Workforce:
- 1-2 full weeks on average

Conclusion and Future Work

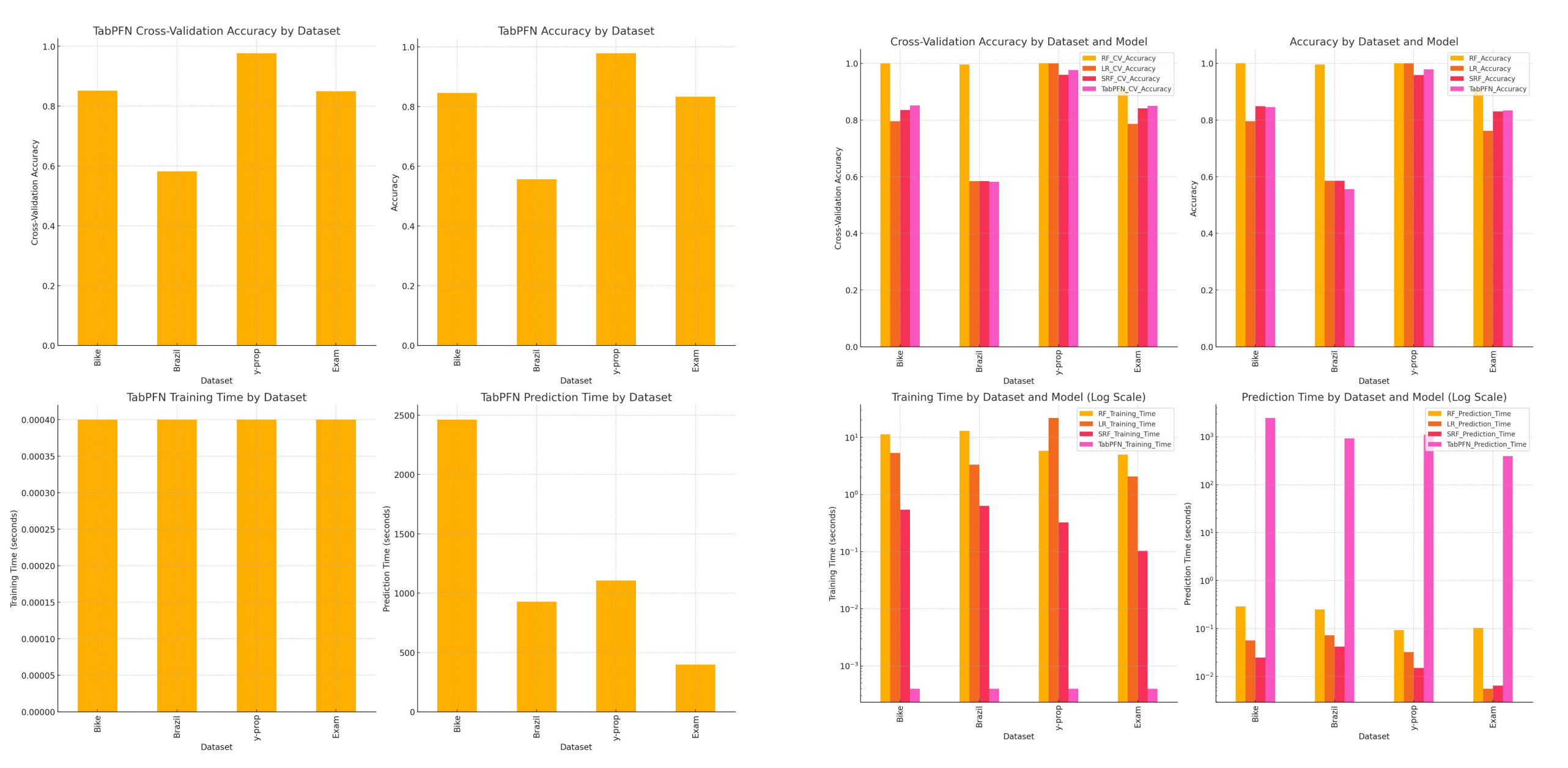
Conclusion:
The AutoML pipeline developed is a comprehensive integration of multiple frameworks, resulting in the attainment of strong performance with optimized ensemble models. It effectively demonstrated the importance of ensemble learning and robust evaluation methods.

Future Work:
1. Exploring additional AutoML frameworks and techniques to further enhance the pipeline's capabilities.
2. Incorporating advanced feature engineering and data augmentation methods to improve model performance.
3. Extending the solution to handle various machine learning tasks, including classification and clustering.
4. Investigating the use of meta-learning to iteratively enhance the efficacy of the models developed.
5. Using multifidelity optimization for time to r2-score on the different tools.

Number of queries for test score generation: **1**

Work with TabPFN

TabPFN provides competitive accuracy and extremely fast training times but at the cost of slower prediction times. This makes it potentially very useful for scenarios where prediction time is less critical or where the model needs to be retrained frequently. However, for applications requiring rapid predictions, other models might be more suitable in these cases where these datasets are converted into classification problems.



References

- **AutoGluon :** https://auto.gluon.ai/stable/index.html
- **Auto-Sklearn:** https://automl.github.io/auto-sklearn/master/
- **FLMAL**: https://microsoft.github.io/FLAML/docs/Getting-Started/
- **TPOT**: https://epistasislab.github.io/tpot/
- **MLJAR**: https://mljar.com/docs/
- **TabPFN:** https://www.automl.org/tabpfn-a-transformer-that-solves-small-tabular-classification-problems-in-a-second/

AutoML.org