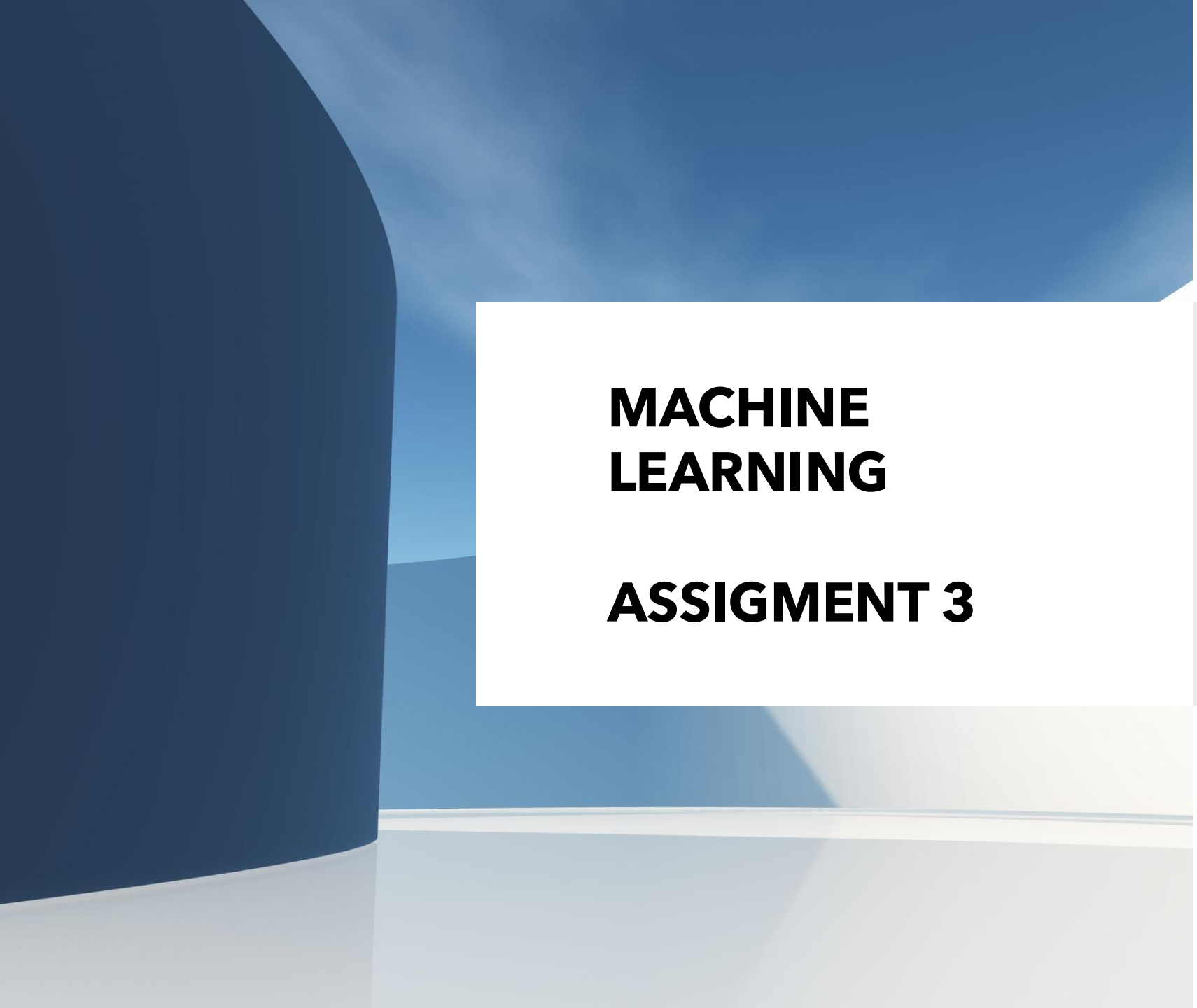


MACHINE LEARNING

ASSIGMENT 3

AUTOMATED
MACHINE
LEARNING

Datasets

- Cars dataset
  Data frame of the suggested retail prices (column Price) and various characteristics of each car.
  - 804 instances x 18 features
  - Target variable: Car Price

- Employee salaries dataset
  Annual salary information including gross pay and overtime pay for all active, permanent employees of Montgomery County, MD paid in calendar year 2016.
  - 9228 instances x 13 features
  - Target variable: Current annual salary

Datasets

- Energy efficiency dataset
  This dataset looked into assessing the heating load and cooling load requirements of buildings (that is, energy efficiency) as a function of building parameters.
  - 768 instances x 10 features
  - Target variable: Heating Load

- Toronto rental dataset
  The Toronto Apartment Rental prices from various sources in local websites.
  - 1124 instances x 7 features
  - Target variable: Rental Price

Simulated Annealing

- Custom implementation
- Supports 5 different regression models implemented with PyTorch:
  - FNNRegressor
  - LightGBMRegressor
  - LogisticRegressor
  - MLPRegressor
  - XGBoostRegressor
- Attempts to find the optimal model and hyperparameter setup for the given dataset

Simulated Annealing Models

FNNRegressor (Feedforward Neural Network)

Deep Learning

Captures complex patterns, good for large datasets but, needs tuning.

LightGBMRegressor

Gradient Boosting

Fast, efficient on large datasets, handles categorical features well.

LogisticRegressor

Linear Model

Simple, interpretable, effective for binary classification.

MLPRegressor (Multilayer Perceptron)

Neural Network

Learns nonlinear relationships, requires careful hyperparameter tuning.

XGBoostRegressor

Gradient Boosting

High accuracy, handles missing values, robust to overfitting.

Simulated annealing Implementation

- Caller provides a list of dataclasses defining the available regression models and their hyperparameters

```
class ModelConfig:
  model_cls: Type[BaseRegressor]
  parameters: dict[str, list[any]]
  ...
```

- Algorithm generates a new solution on each iteration
  - New solution uses a different regression model with chance `p_test_different_model`
  - Otherwise, we select a random hyperparameter of the current regresiossion model and set a new value within `neighborhood_range`
- New solutions are accepted if either
  - They score better in terms of RMSE
  - `random() < math.exp(current_rmse – neighbor_rmse / self._temperature))`

Simulated annealing Implementation

- Evaluation
  - We calculate RMSE and R^2 for each solution
  - New solutions are selected with the goal of minimizing RMSE

- Hyperparameters
  - Parameters implemented per techique
    - `FNNRegressor`: epochs, batch size, learning rate
    - `LightGBMRegressor`: boost rounds
    - `LogisticRegressor`: epochs, batch size, learning rate
    - `MLPRegressor`: epochs, batch size, learning rate
    - `XGBoostRegressor`: boost rounds

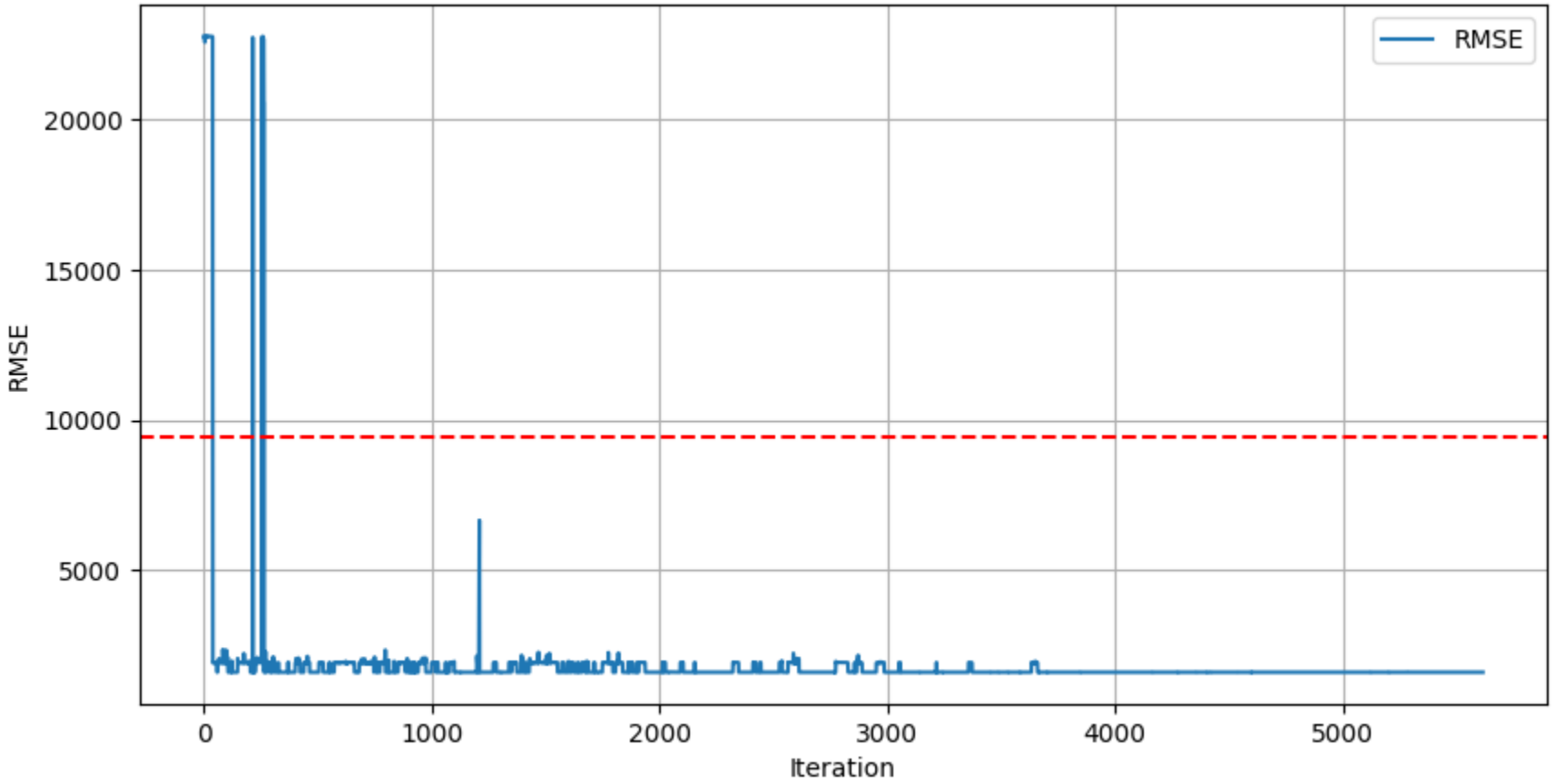
Simulated annealing Implementation

- Halting criteria
  - Max steps
  - Max time
  - Min temperature
    - Alternatively: Reheat on reaching min temperature and continue running

- Other parameters for our implementation of Simulated Annealing
  - Initial temperature
    - Alternatively: Provide initial acceptance rate and let the algorithm calculcate initial temperature based on that
  - Cooling factor
  - Iterations per step (iterations before decreasing temperature)
  - Previously mentioned: `neighborhood_range`, `p_test_different_model`

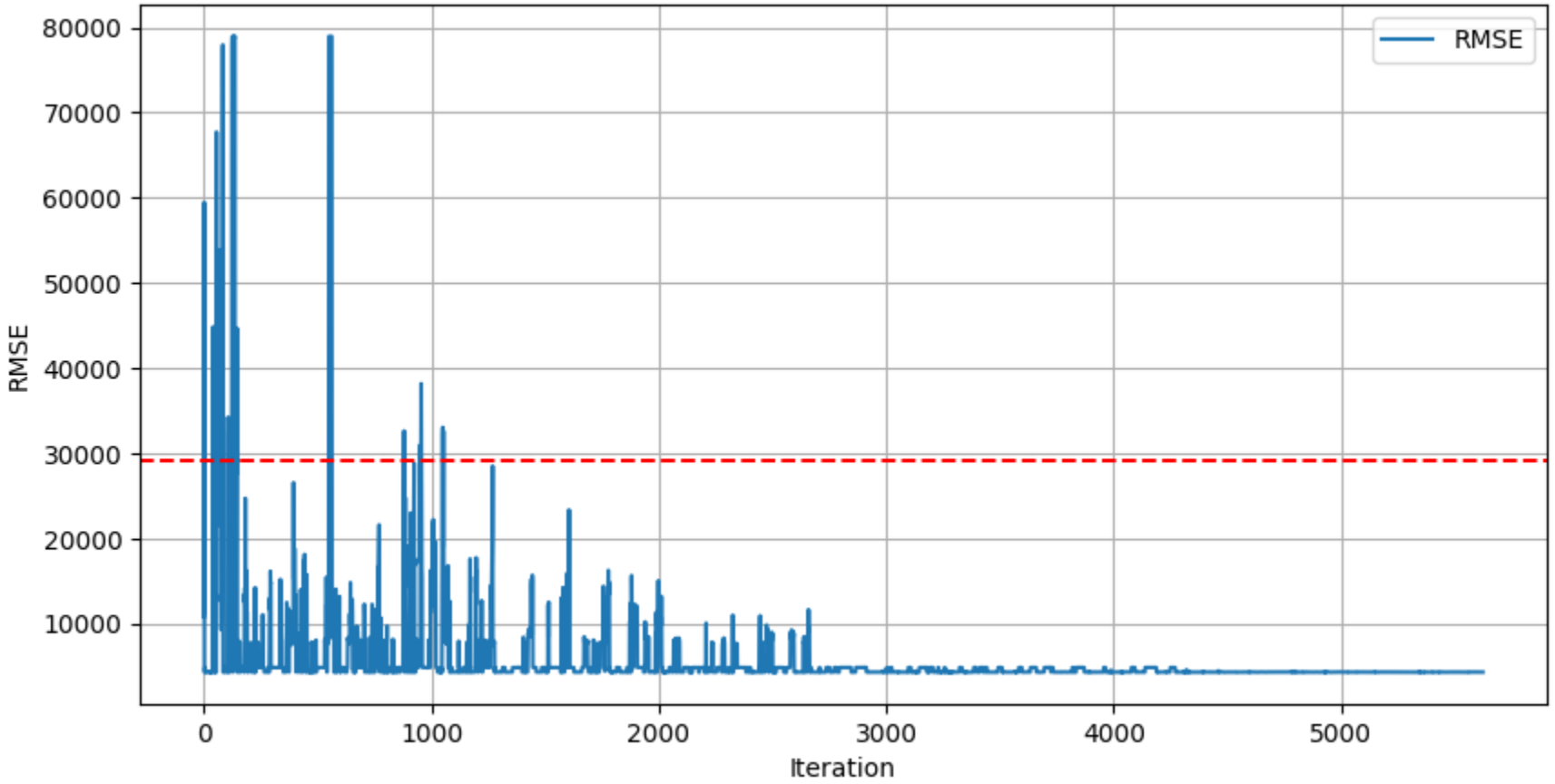
Simulated annealing Results: Cars

- Model: XGBoost Regressor, `num_boost_round`: 320
- RMSE: 1603.64 (Std. Dev.: 9436.03)
- R2: 0.9709



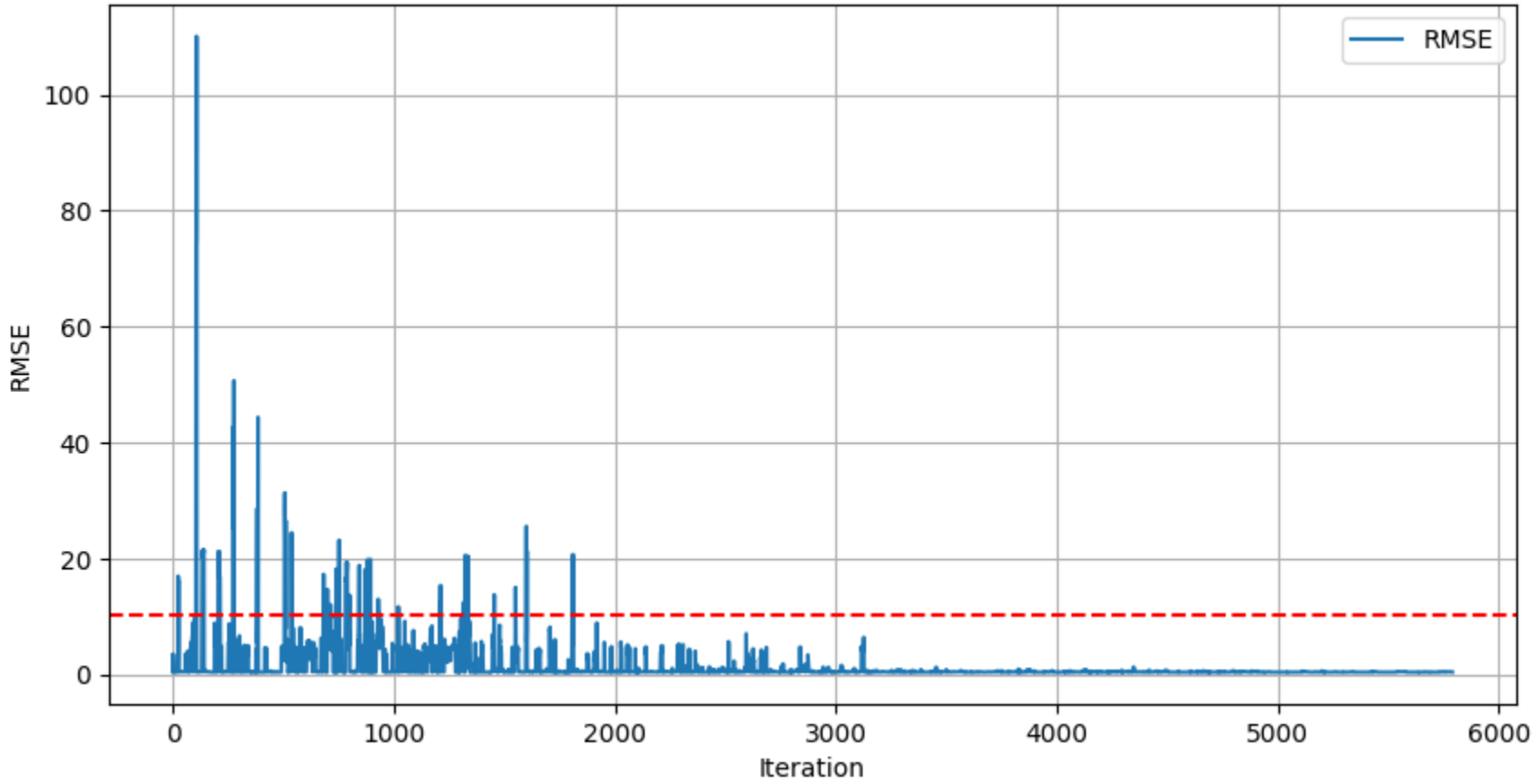
Simulated annealing Results: Employee Salaries

- Model: LightGBM Regressor, `num_boost_round`: 493
- RMSE: 4334.54 (Std. Dev.: 29199.59)
- R2: 0.978



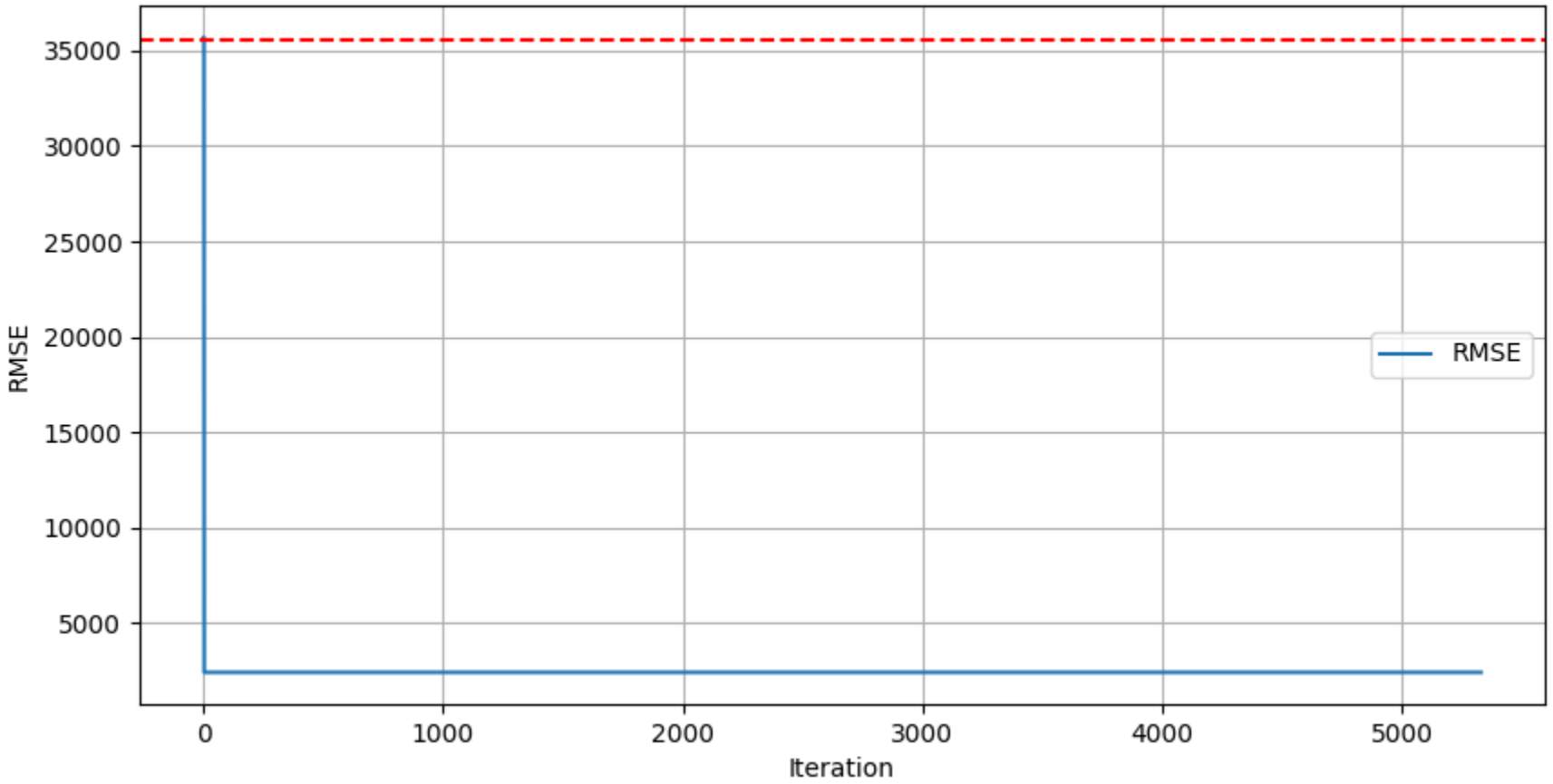
Simulated annealing Results: Energy Efficiency

- Model: XGBoost Regressor, `num_boost_round`: 63
- RMSE: 0.4338 (Std. Dev.: 10.3167)
- R2: 0.9982



Simulated annealing Results: Toronto Rental

- Model: XGBoost Regressor, `num_boost_round`: 55
- RMSE: 2398.76 (Std. Dev.: 35594.22)
- R2: 0.9954



State of the art AutoML systems

- Using AutoGluon & FLAML

- AutoGluon
  - Open-source AutoML toolkit developed by Amazon (AWS AI)
  - Train and deploy high-accuracy machine learning and deep learning models on image, text, time series, and tabular data …
  - Hyperparameter optimization based on random search methods

- FLAML
  - A Fast and Lightweight AutoML Library
  - Open-source library developed by Microsoft
  - Cost-frugal optimization, BlendSearch

Code Architecture Overview

- Dataset Handling:
  - Data is loaded either from OpenML or from locally saved CSV files.
  - Multiple datasets (cars, employee salaries, energy efficiency, Toronto rental) are prepared with dedicated functions.
- Preprocessing:
  - Uses scikit-learn pipelines with steps like standard scaling, one-hot encoding, ordinal encoding, and missing value imputation.
- Timing & Utilities:
  - A decorator (`timer`) is used to measure the execution time of each dataset preparation function. • Helper functions convert data into NumPy arrays for model compatibility.

Dataset Preparation

- Functions for Each Dataset:
  - `prepare_cars_dataset`
  - `prepare_employee_salaries_dataset`
  - `prepare_energy_efficiency_dataset`
  - `prepare_toronto_rental_dataset`
- Key Points:
  • Each function loads the dataset, applies preprocessing transformations, and splits data into training and testing sets. • A subset of data is used during test runs to ensure faster execution.

The Two AutoML Frameworks

- AutoGluon
- What is AutoGluon?
  • A robust library designed for tabular data, image, text, and more.
- In the Code:
  • Uses `TabularPredictor` to automatically train and tune models. • Retrieves a leaderboard to identify the best performing model.
- Evaluation:
  • Predictions are made on the test set and evaluated with R^2 and RMSE.
- FLAML
- What is FLAML?
  • A lightweight, efficient AutoML library focused on low computational overhead.
- In the Code:
  • Uses a time budget (300 seconds) to search for the best estimator. • Directly fits the model and evaluates performance using the same metrics (R^2 and RMSE).

Training & Evaluation Process

- Process Overview:
- Training:
  - Both frameworks are used on the same preprocessed dataset.
- Model Selection:
  - The best model is chosen based on higher R^2 (goodness-of-fit) and lower RMSE (error measure).
- Result Aggregation:
  - Results from each dataset (including best algorithm, R^2, and RMSE) are stored in a CSV file (`automl_results.csv`).
- Code Flow:
  - The function `train_on_all_datasets()` loops through the prepared datasets, applies both frameworks, and compares their performance.

Interpreting the Results

- Metrics Explained:
- R^2 Score:
  - Measures the proportion of variance explained by the model.
  - Higher values indicate better performance.
- RMSE (Root Mean Squared Error):
  - Measures the average magnitude of prediction errors.
  - Lower values indicate better performance.
- Outcome:
  - For each dataset, the framework that achieves the highest R^2 and the lowest RMSE is considered the best. • This comparison highlights which AutoML solution works best depending on the dataset characteristics

```
Dataset,Best Algorithm,R2,RMSE
toronto_rental,extra_tree,0.907792917478323 6,145.50102054514483
employee_salaries,lgbm,0.8992677201385125,12184.397141027124
energy_efficiency,xgboost,0.9943620609304966,0.6615496371394484
cars,WeightedEnsemble_L2,0.9783074218444815,1016.5634358604723
```

Result comparison: Simulated Annealing vs. AutoML

- Cars dataset
  - Simulated Annealing: XGBoost Regressor (320 rounds)
    - R2: 0.9709
  - AutoML (AutoGluon & FLAML): Weighted Ensemble L2
    - R2: 0.9783

- Employee salaries dataset
  - Simulated Annealing: LightGBM Regressor (493 rounds)
    - R2: 0.978
  - AutoML (AutoGluon & FLAML): LGBM
    - R2: 0.8993

Result comparison: Simulated Annealing vs. AutoML

- Energy efficiency dataset
  - Simulated Annealing: XGBoost Regressor (63 rounds)
    - R2: 0.9982
  - AutoML (AutoGluon & FLAML): XGBoost
    - R2: 0.9944

- Toronto rental dataset
  - Simulated Annealing: XGBoost Regressor (55 rounds)
    - R2: 0.9954
  - AutoML (AutoGluon & FLAML): Extra Tree
    - R2: 0.9078

Result comparison: Simulated Annealing vs. AutoML

- Interpretation
  - Simulated annealing and the third-party autoML solutions converged on the same/similar models for two datasets
    - Employee salaries (LightGBM)
    - Energy Efficiency (XGBoost
  - They came up with different solutions for
    - Cars (XGBoost vs. Weighted Ensemble L2)
    - Toronto Rental (XGBoost vs. Extra Tree)

Obvious explanation: we didn't implement those models for our simulated annealing approach.

Although: The XGBoost regressor outscored Extra Tree on Toronto Rental

Result comparison: Simulated Annealing vs. AutoML

- Interpretation
  - Simulated Annealing outscored the third-party solutions on 3 out of 4 datasets.
    The difference in performance was large on 2 of those datasets.

  - Possible explanations:
    - Differences in training/selection time?
    - Number of different models and hyperparameters to test?
    - Simulated annealing vs. Other optimization mechanisms?