Feature Construction

PV211 Seminar on ML, IR, and SV

Zuzana Pitsmausová

25 April, 2024

Outline

Outline

1. Introduction; Motivation
2. Feature Construction & SOTA
3. Background
4. Problem Setting
5. Evolutionary Forest
6. M3GP

Feature

	laptop_ID	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price_euros
0	1	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	1339.69
1	2	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	898.94
2	3	HP	250 G6	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	575.00

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

"A **feature** is an **individual measurable property** within a **recorded dataset**. In machine learning and statistics, features are often called "**variables**" or "**attributes**."

In a **patient medical dataset**, features could be **age**, **gender**, **blood pressure**, **cholesterol level**, and other observed characteristics relevant to the patient.

Introduction; Motivation

- **crucial step** in the ML pipeline
- quality of features **affects model's performance**
- creating new features / transforming existing ones
- growing interest in developing new methods
- increasing complexity and diversity of data sources

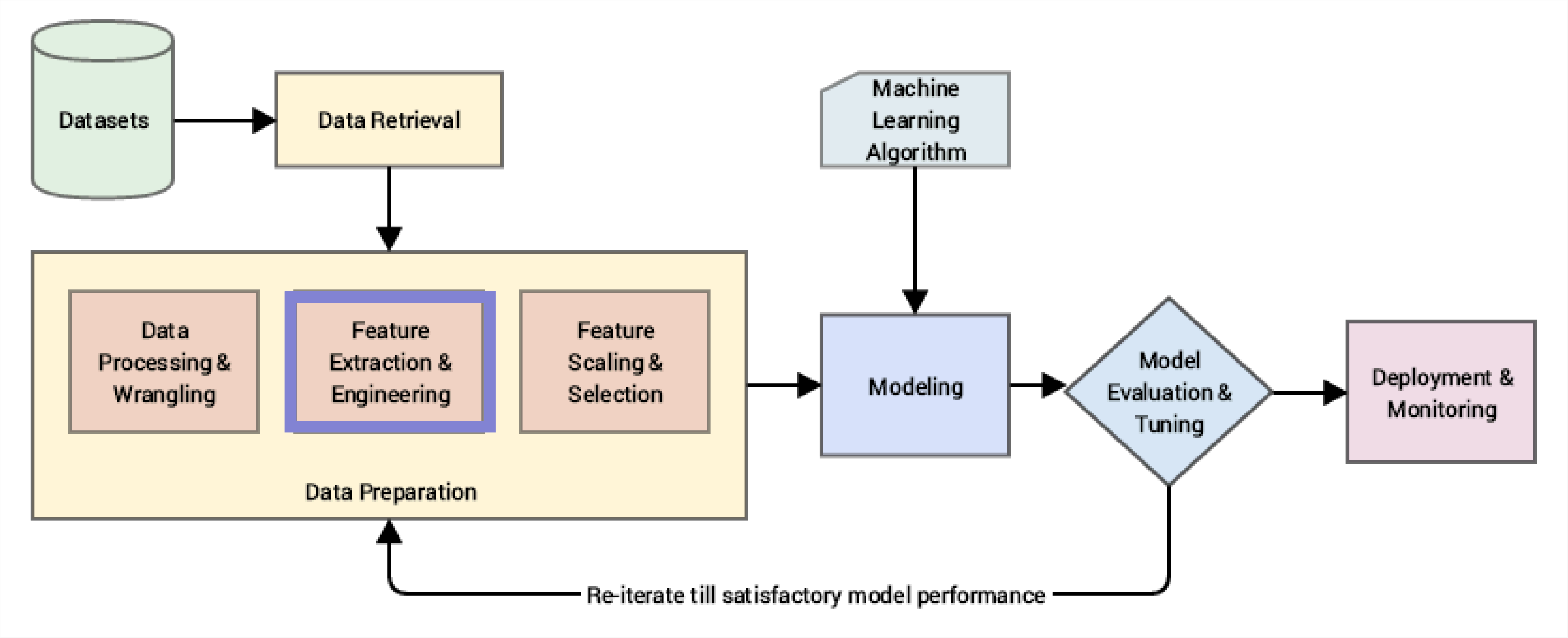


Fig. 0. ML pipeline with highlighted Feature Extraction & Engineering phase

Outline

Feature Construction (FC)

- new features are constructed from raw data or previously constructed features
- **in order to:**
  - improve robustness
  - interpretability
  - and/or generalization
- **result:** features, that may better describe target concept

- **in SOTA - three groups:**
  - standalone (operator-based)
  - embedded into automated ML
  - feature interactions

Standalone FC Methods

- **following steps:**
  - (1) generating a set of candidates
  - (2) ranking candidate features
  - (3) evaluating and selecting high-ranked features
  - (4) adding promising features to dataset

- mostly **human-guided**, **less efficient**
- problem- or domain-specific

Embedded FC Methods

- **automated** construction of features
- reduces exploration time,
- two approaches
  - **expand-reduce** (ExploreKit, Deep Feature Synthesis, Auto-Sklearn)
  - **wrapper approach**
- usually not fully automated

Feature Interactions

- certain features are **not individually** related to target concept
- need to be **combined** with other features
- to avoid construction of **meaning-less** features
- **before FC process**

Outline

1. Introduction; Motivation
2. Feature Construction & SOTA
3. **Background**
4. Problem Setting
5. Evolutionary Forest
6. M3GP

Genetic Programming

- one of the **evolutionary computation** (EC) algorithms
  - **other:** ant colony opt. (ACO), swarm opt. (SO)
- evolution of computer programs
- **representation:** tree-like structures
- **individuals:** treated as trees
- **operators:** mutation, crossover

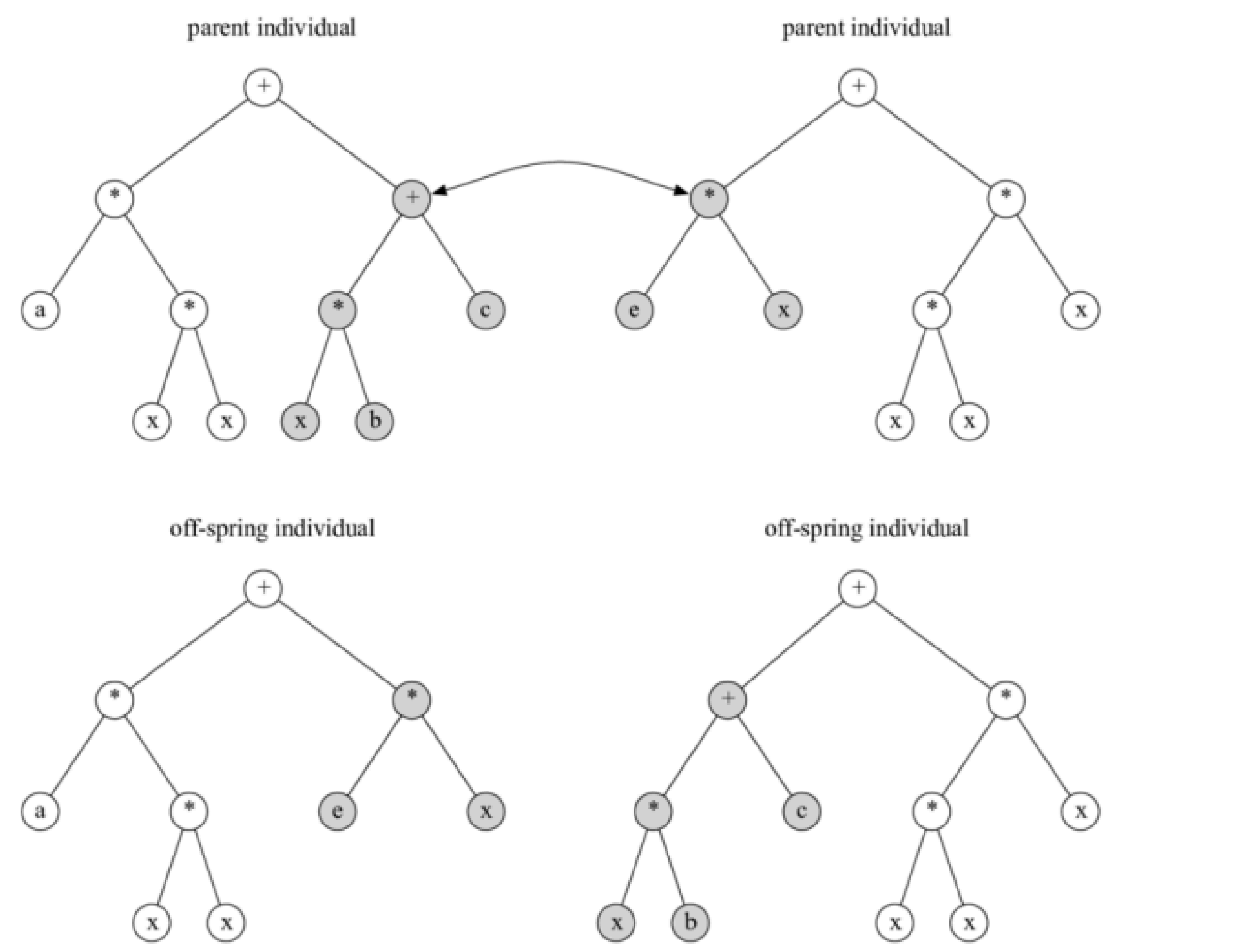


Fig. 1. GP trees – parent individuals and off-spring individuals. Figure is showing one of the genetic operators – crossover. [7]

Genetic Programming

Evolutionary algorithm

generate(P_0); *(generate initial population)*
$t = 0$;
while not termination-criterion($P(t)$) **do**
  evaluate(P_t);
  P'_t = selection(P_t); *(selection strategy)*
  P'_t = reproducion(P'_t); *(reproduction strategy)*
  evaluate(P'_t);
  P_{t+1} = replace(P_t, P'_t); *(replacement strategy)*
  $t = t + 1$;
end while;
output: best individual or best population found

Fig. 2. Evolutionary algorithms: base template

Outline

Pipeline

1. Pick two frameworks for FC based on GP
    a. **Evolutionary Forest**
    b. **M3GP**
2. Pick four datasets
    a. **toy ds:** diabetes
    b. **kaggle ds:** laptop/house prices, medical insurance
3. Datasets preprocessing

Pipeline

4. Use picked frameworks for creating new features
5. Compare several regressors trained on both old and newly created features (R2)
    a. **AdaBoostReg**
    b. **ExtraTreeReg**
    c. **RandomForestReg**
6. Plot the results; Compare picked frameworks

Datasets

- **toy dataset:** [diabetes]
- **three kaggle datasets:** Laptop Prices, House Prices, Medical Insurance
- **basic preprocessing:**
  - check for NaN values
  - check distribution of particular cols
  - if necessary encoding features to numerical form
  - additional small adjustments
- train, test split (80:20)

	Company int64	TypeName int64	Inches int64	ScreenResolution i...	Ram int64	OpSys int64	Weight int64	Price_euros float64	Memory_Type int64	Memory_gb int64	Gpu_type int64	Cpu_type int64
0	1	4	7	10	3	8	37	1339.69	5	4	9	12
1	1	4	7	1	3	8	34	898.94	0	4	6	12
2	7	3	14	3	3	4	72	575.0	5	7	6	15
3	1	4	13	12	5	8	69	2537.45	5	10	3	12
4	1	4	7	10	3	8	37	1803.6	5	7	9	12

	price float64	crime_rate float64	resid_area float64	air_qual float64	room_num float64	age float64	dist1 float64
0	24.0	0.00632	32.31	0.538	6.575	65.2	4.35
1	21.6	0.02731	37.07	0.469	6.421	78.9	4.99
2	34.7	0.02729	37.07	0.469	7.185	61.1	5.03
3	33.4	0.03237	32.18	0.458	6.998	45.8	6.21
4	36.2	0.06905	32.18	0.458	7.147	54.2	6.16

	age int64	sex object	bmi float64	children int64	smoker object	region object
0	19	female	27.9	0	yes	southwest
1	18	male	33.77	1	no	southeast
2	28	male	33.0	3	no	southeast
3	33	male	22.705	0	no	northwest
4	32	male	28.88	0	no	northwest

Frameworks

Evolutionary Forest

pypi v0.2.3

build unknown

docs failing

pyup unknown

An open source python library for automated feature engineering based on Genetic Programming

jespb / Python-M3GP Public

An easy-to-use scikit-learn inspired implementation of the Multidimensional Multiclass Genetic Programming with Multidimensional Populations (M3GP) algorithm

Technology Used

- Python 3.9
- deepnote
- frameworks: **M3GP, EF**

```python
# Necessary imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
# necessary imports
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
from sklearn.datasets import load_diabetes
from sklearn.ensemble import ExtraTreesRegressor, AdaBoostRegressor, GradientBoostingRegressor, RandomForestRegressor
from sklearn.metrics import r2_score
```

Outline

Evolutionary Forest Framework

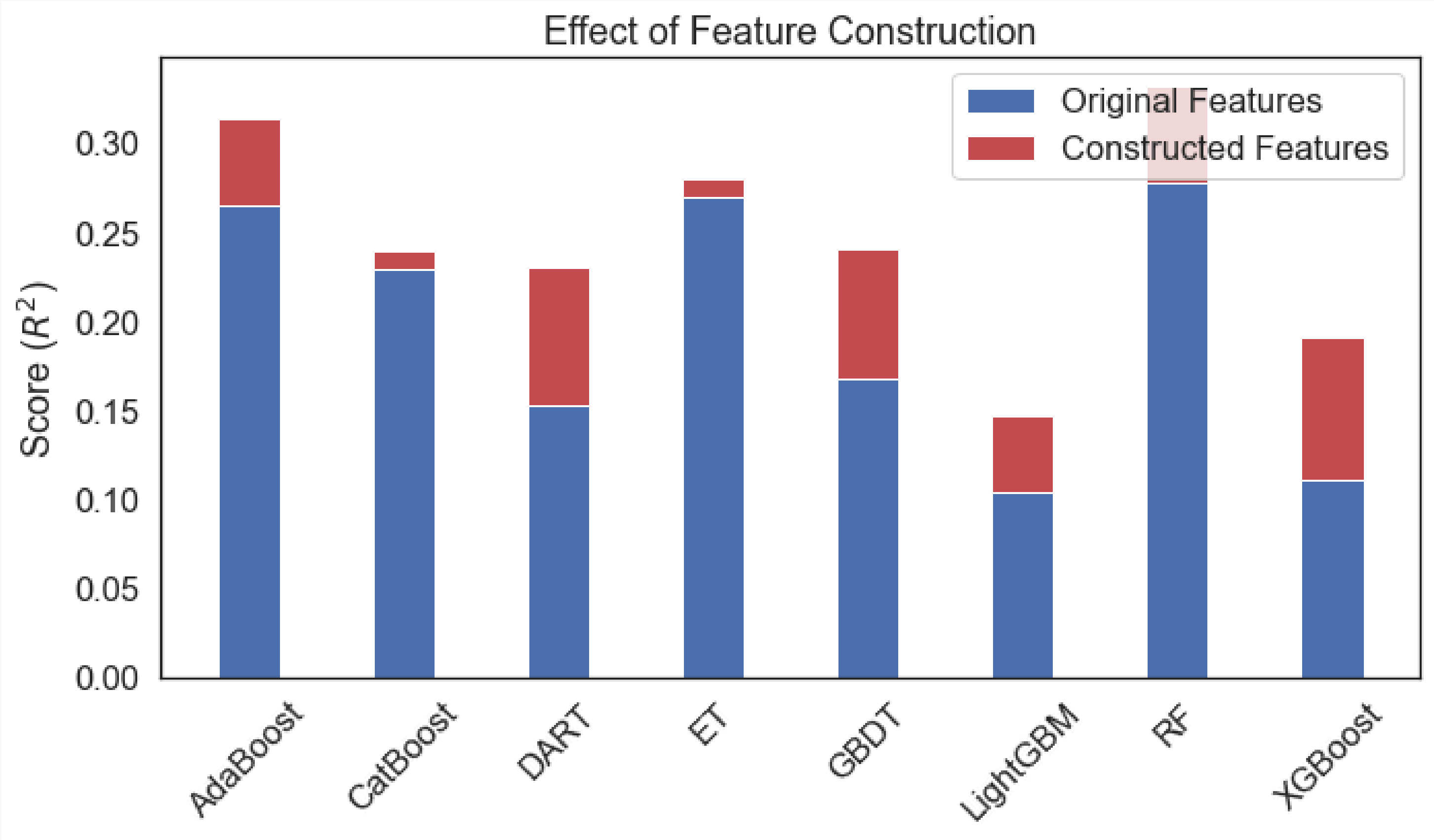
- automated feature engineering library based on GP
- provides ensemble EvolutionaryTreeRegressor model
- new candidate GP individuals generated through
  - mutation
  - crossover
- **each individual:** group of GP trees, one tree represents one feature

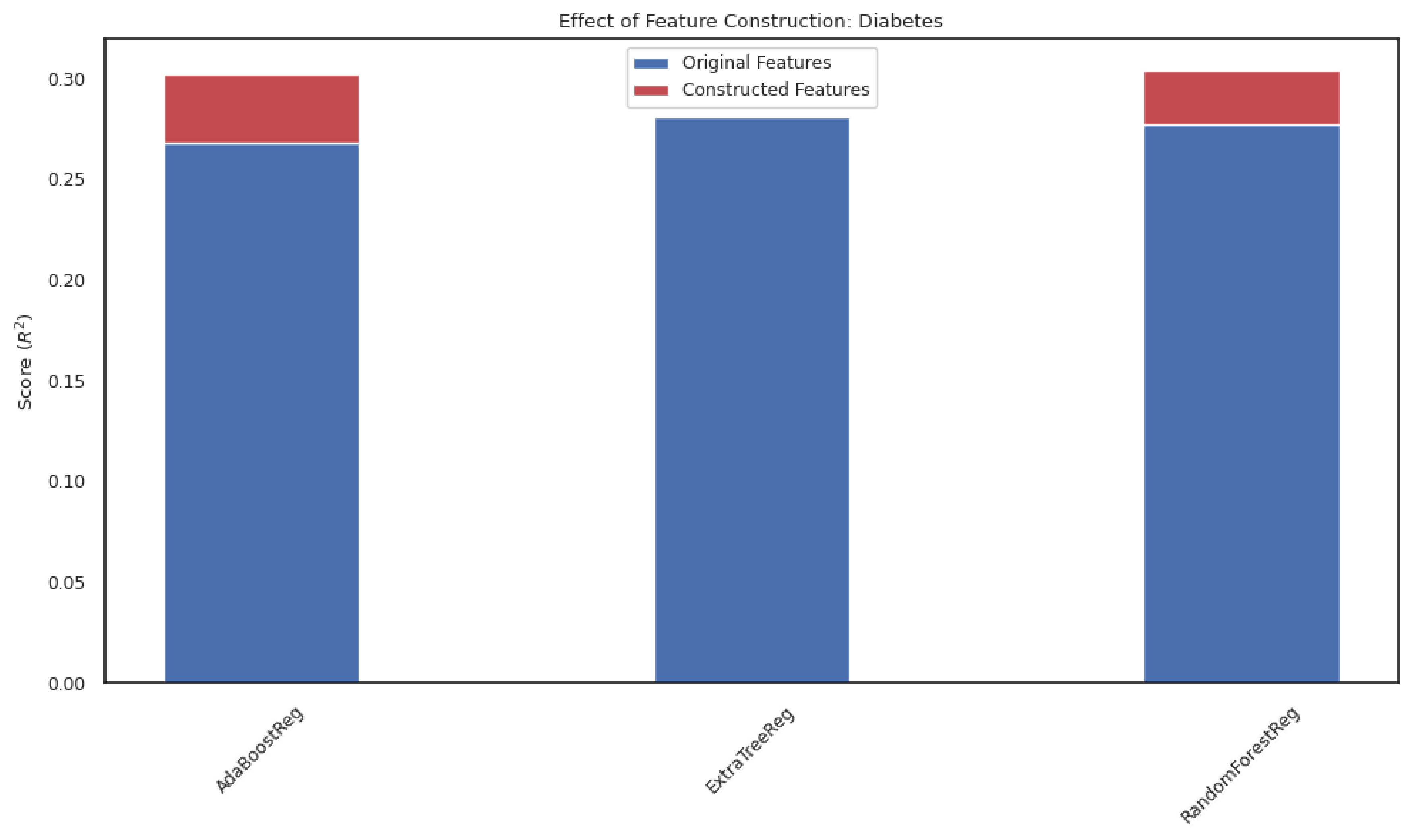
- **option to:**
  - see important features (based on fitness fc) and plot them
  - use them for creating new train/test data
- vast number of hyperparameters (tried adjusting fundamental ones)

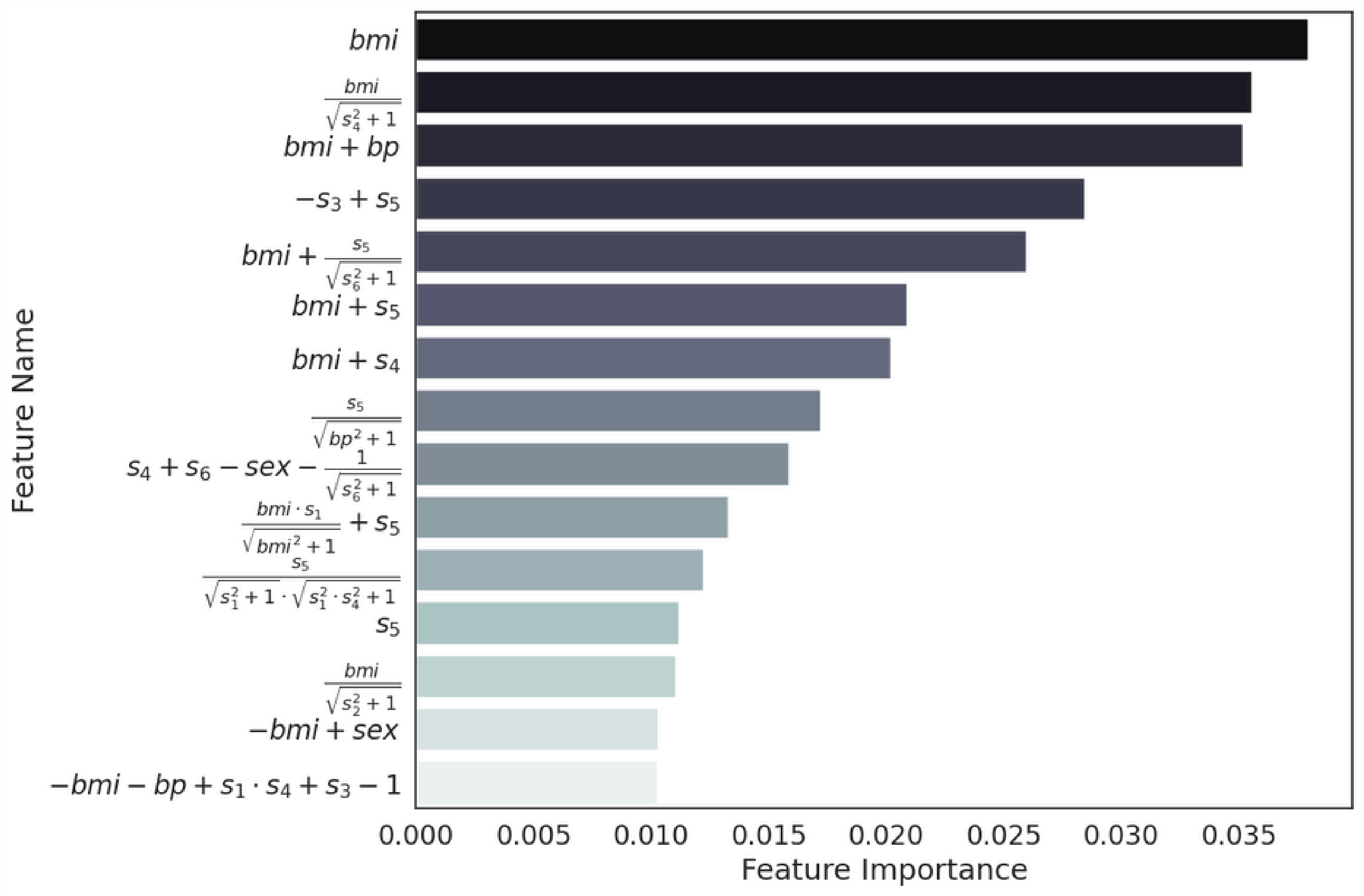
```python
def __init__(self,
             # Basic GP Parameters (Core)
             n_pop=50,  # Population size
             n_gen=20,  # Number of generations
             cross_pb=0.5,  # Probability of crossover
             mutation_pb=0.1,  # Probability of mutation
             max_height=8,  # Maximum height of a GP tree
             min_height=0,  # Minimum height of a GP tree
             gene_num=5,  # Number of genes in each GP individual
             mutation_scheme='uniform',  # Mutation scheme used in GP
             verbose=False,  # Whether to print verbose information
             basic_primitives=True,  # Primitive set used in GP
             normalize=True,  # Whether to normalize before fitting a model
             select='AutomaticLexicaseFast',  # Name of the selection operator
             elitism=0,  # Number of the best individuals to be directly passed to next generation

             # Basic GP Parameters (Not Recommend to Change)
             external_archive=None,  # External archive to store historical best results
             original_features=False,  # Whether to use original features in the model or not
             second_layer=None,  # Strategy to induce a second layer for assigning different weights in ensemble
             allow_revisit=False,  # Whether to allow repetitive individuals
             n_process=1,  # Number of processes for parallel execution
             constant_type='Int',  # Type of constants used in GP
             early_stop=-1,  # Early stopping criteria (number of generations)
             random_fix=True,  # Whether to use random fix when height limit is not satisfied in GP

             # Ensemble Learning Parameters
             base_learner='Random-DT',  # Base learner used in the ensemble
             min_samples_leaf=1,  # Minimum samples required to form a leaf node in a tree
             max_tree_depth=None,  # Maximum depth of a decision tree
             cv=5,  # Number of cross-validation folds
             score_func='R2',  # Scoring function for evaluation
             ensemble_prediction='Mean',  # Method of ensemble prediction
             ensemble_selection=None,  # Ensemble selection method
             ensemble_size=100,  # Size of the ensemble model

             # Soft PS-Tree Parameters
             partition_number=4,  # Number of partitions in PS-Tree
             ps_tree_local_model='RidgeCV',  # Type of local model used in PS-Tree
             dynamic_partition='Self-Adaptive',  # Strategy to dynamically change partition scheme
             ps_tree_cv_label=True,  # Whether to use label information in CV in PS-Tree
             ps_tree_partition_model='DecisionTree',  # Type of model used for partitioning
             only_original_features=True,  # Whether to only use original features in PS-Tree
             shared_partition_scheme=False,  # Whether to use shared partition scheme for all individuals
             max_leaf_nodes=None,  # Maximum height of each decision tree

             # SR-Forest Parameters (TEVC 2023)
             ps_tree_ratio=0.1,  # Ratio of PS-Tree in multi-fidelity evaluation
             decision_tree_count=0,  # Number of piecewise trees in a SR-Tree

             # More Parameters
             initial_tree_size=None,  # Initial size of GP tree
             basic_gene_num=0,  # Number of basic genes in a MGP
             clearing_cluster_size=0,  # Cluster size in clearing
             reduction_ratio=0,  # Ratio of samples removed in pre-selection based on filters
             random_state=None,  # Random state used for reproducibility
             validation_size=0,  # Size of the validation set for using in HOF
             mab_parameter=None,  # Parameters for the MAB
             interleaving_period=0,  # Period of interleaving (Multi-fidelity Evaluation)

             # MEGP Parameters (EuroGP 2023)
             map_elite_parameter=None,  # Hyper-parameters for MAP-Elite

             # EvoFeat Parameters
             class_weight=None,  # Weight for each class in multi-class classification

             # Deprecated parameters
             boost_size=None,  # Alias of "ensemble_size"
             semantic_diversity=None,  # Alias of "ensemble_selection"

             # MGP hyperparameters
             mgp_mode=False,  # Whether to use MGP
             mgp_scope=None,  # Scope of MGP
             number_of_register=10,  # Number of registers in MGP
             intron_probability=0,  # Probability of intron gene in MGP
             register_mutation_probability=0.1,  # Probability of register mutation in MGP
             delete_irrelevant=False,  # Whether to delete irrelevant genes in MGP
             delete_redundant=False,  # Whether to delete redundant genes in MGP

             # Semantic Variation (Trail-and-error to generate new trees)
             semantic_variation=False,  # Whether to use semantic variation in GP
             correlation_threshold=None,  # Correlation threshold for pre-selection
             correlation_mode=None,  # Correlation mode for pre-selection

             # MGP hyperparameters
             irrelevant_feature_ratio=0.01,  # Ratio of irrelevant features in MGP
             strict_layer_mgp=True,  # Whether to use strict layering in MGP
             number_of_parents=0,  # Number of parents in crossover
             # Strategies
             bloat_control=None,  # Bloat control method in GP

             # SHM-GP hyperparameters
             intron_gp=False,  # Whether to use intron GP

             # User Parameters
             custom_primitives=None,  # Custom primitives for GP

             # Debug Parameters
             test_fun=None,  # Test function for evaluation

             # Experimental Parameters (Maybe deprecated at any version)
             diversity_search='None',  # Strategy to assign diversity objective
             bootstrap_training=False,  # Whether to use bootstrap samples for training
             mean_model=False,  # Whether to use mean model for predictions
             environmental_selection=None,  # Environmental selection method
             pre_selection=None,  # Pre-selection method
             eager_training=False,  # Whether to train models eagerly
             useless_feature_ratio=None,  # Ratio of useless features to be removed
             weighted_coef=False,  # Whether to use weighted coefficients
             feature_selection=False,  # Whether to perform feature selection
             outlier_detection=False,  # Whether to perform outlier detection
             semantic_repair=0,  # Semantic repair method
             dynamic_reduction=0,  # Dynamic reduction strategy
             active_gene_num=0,  # Number of active genes in MGP
             intron_threshold=0,  # Threshold for identifying introns in MGP
             force_sr_tree=False,  # Whether to force use SR-Tree
             gradient_boosting=False,  # Whether to use gradient boosting
             post_prune_threshold=0,  # Threshold for post-pruning of features in GP
             redundant_hof_size=0,  # Size of redundant Hall of Fame
             delete_low_similarity=False,  # Whether to delete low similarity genes
             importance_propagation=False,  # Whether to use importance propagation in GP
             ridge_alphas=None,  # Alpha values for Ridge regression
             parsimonious_probability=1,  # Probability of GP with parsimonious terminal usage
             shared_eda=False,  # Whether to use shared estimation of distribution in GP

             rmp_ratio=0.5,  # Multi-task Optimization
             **params):
```

EF: Diabates

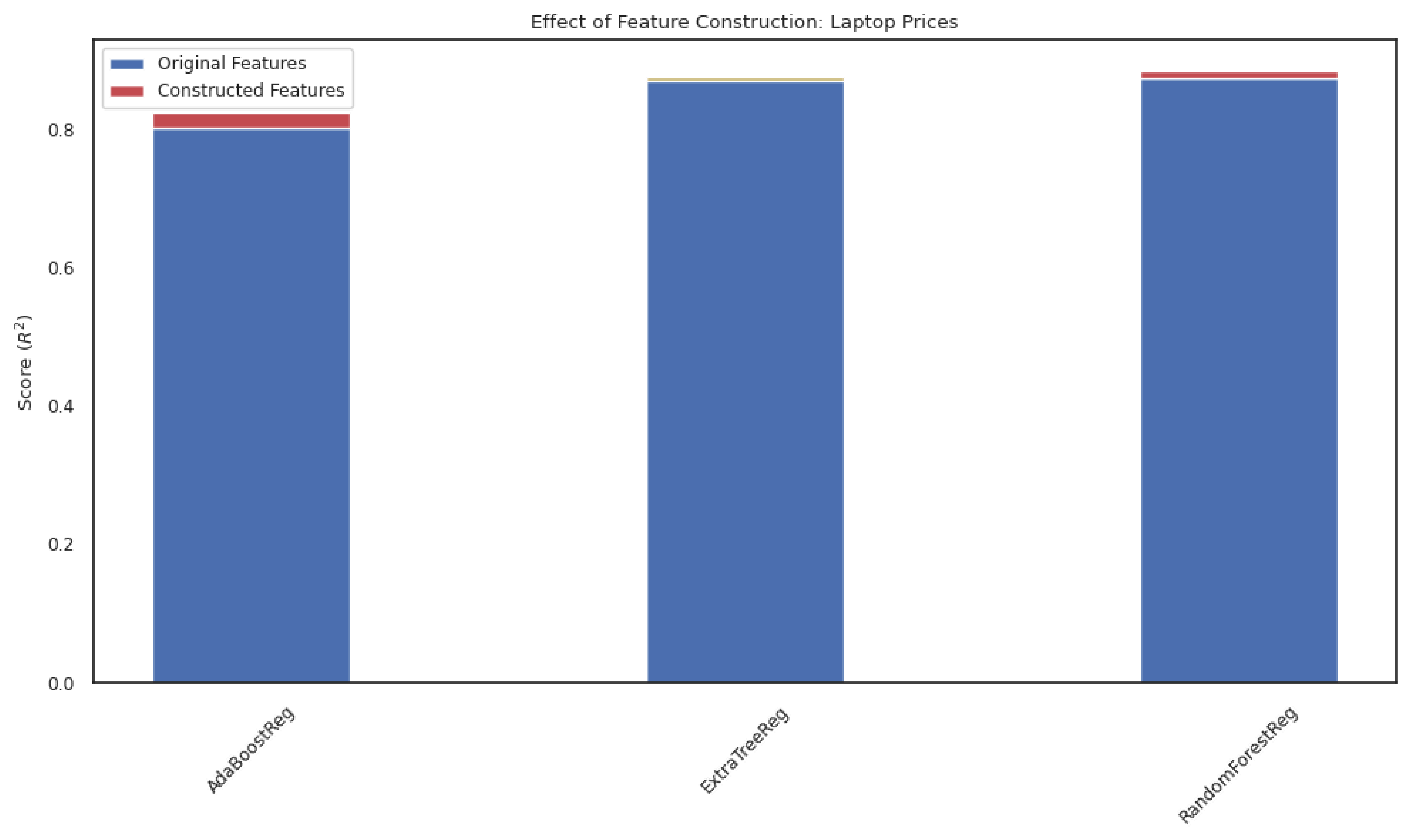
```
Results Diabetes:
                  original_score  new_score  improvement
algorithm
AdaBoostReg             0.267491   0.302103     0.034612
ExtraTreeReg            0.281557   0.280195    -0.001362
RandomForestReg         0.276974   0.304149     0.027175
```







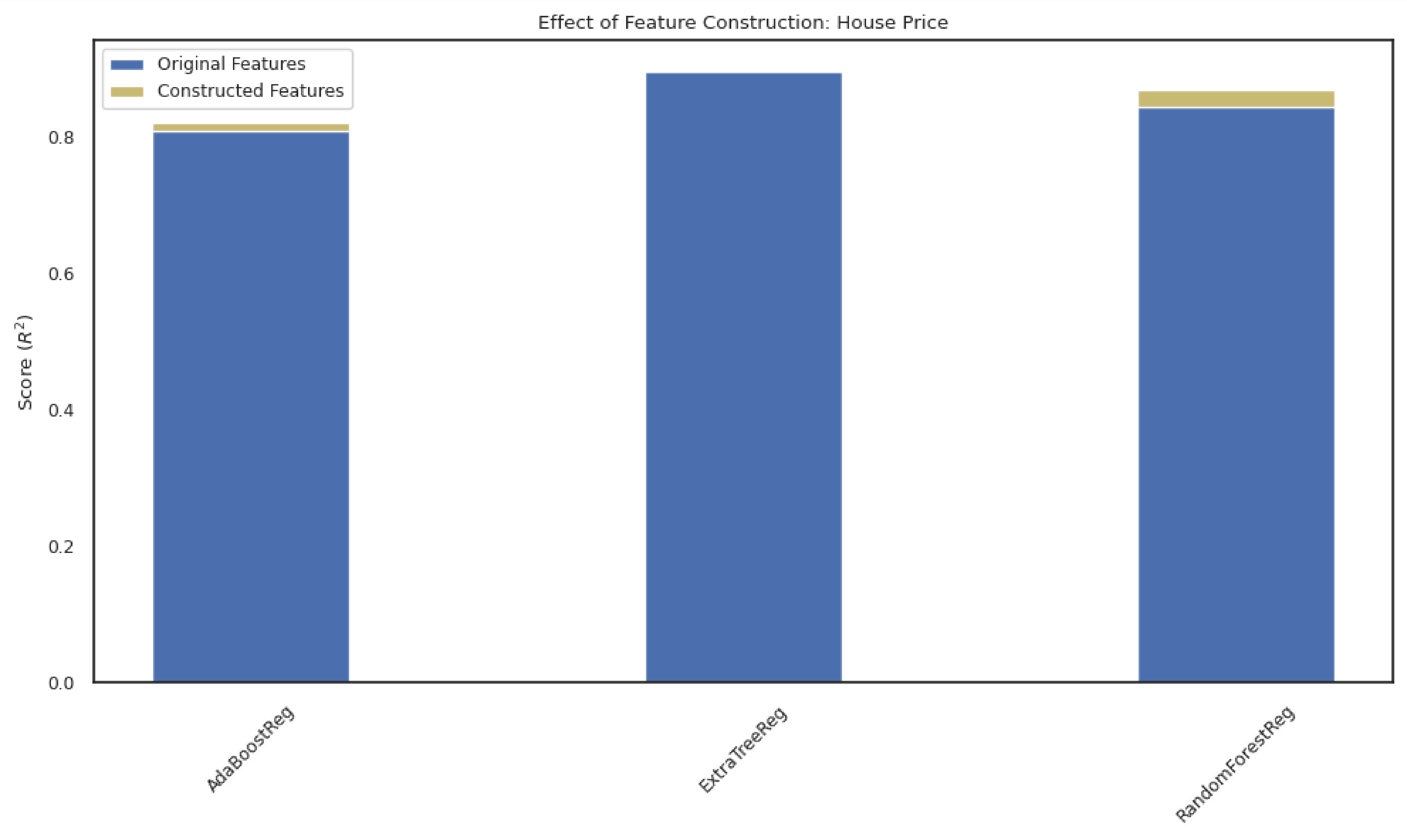
EF: Laptop Prices



Results Laptop Prices:

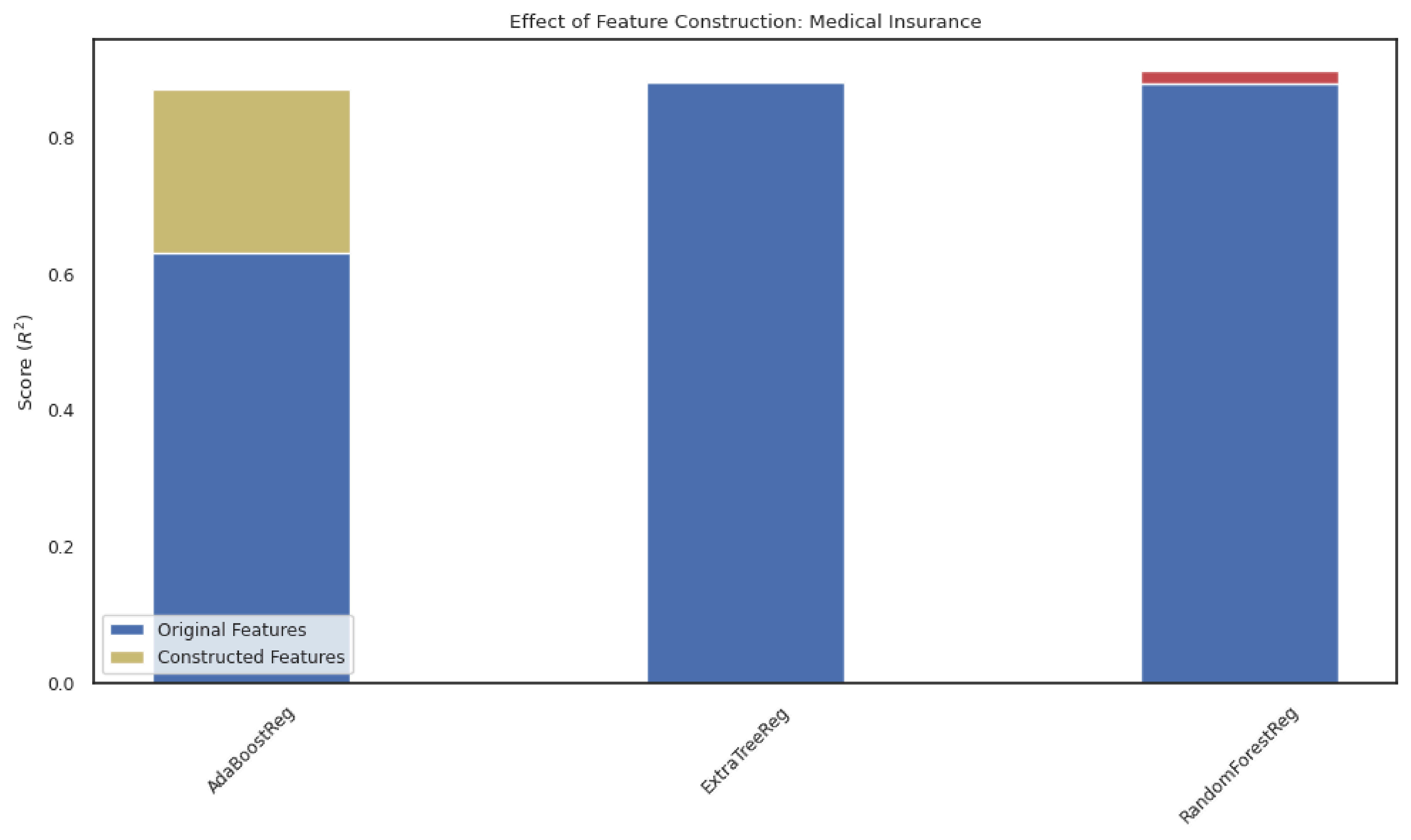
algorithm	original_score	new_score	improvement
AdaBoostReg	0.800832	0.824252	0.023420
ExtraTreeReg	0.873943	0.868466	-0.005478
RandomForestReg	0.873195	0.883751	0.010557

EF: House Prices




```
Results House Price:
                original_score  new_score  improvement
algorithm
AdaBoostReg           0.822125   0.808548    -0.013577
ExtraTreeReg          0.895723   0.897025     0.001302
RandomForestReg       0.869608   0.844354    -0.025254
```

EF: Medical Insurance



Results Medical Insurance:

algorithm	original_score	new_score	diff
AdaBoostReg	0.870117	0.630041	-0.240076
ExtraTreeReg	0.880191	0.883264	0.003073
RandomForestReg	0.878900	0.898728	0.019827

EF: Summary

- tested on four datasets
- hard to jump to certain conclusions
- diverse results
- powerful performance on a small dataset

Outline

M3GP

- originally designed to perform **multiclass classification**
- according to the authors - **powerful FC tool**
- in the original articles - tested within classification task

method:

- feature represented as a tree with:
  - inner nodes (operators)
  - leaves (original features)
- each individual consists of few features (trees)
- three mutation operators, two crossover operators
- support for various fitness functions (reg/class)

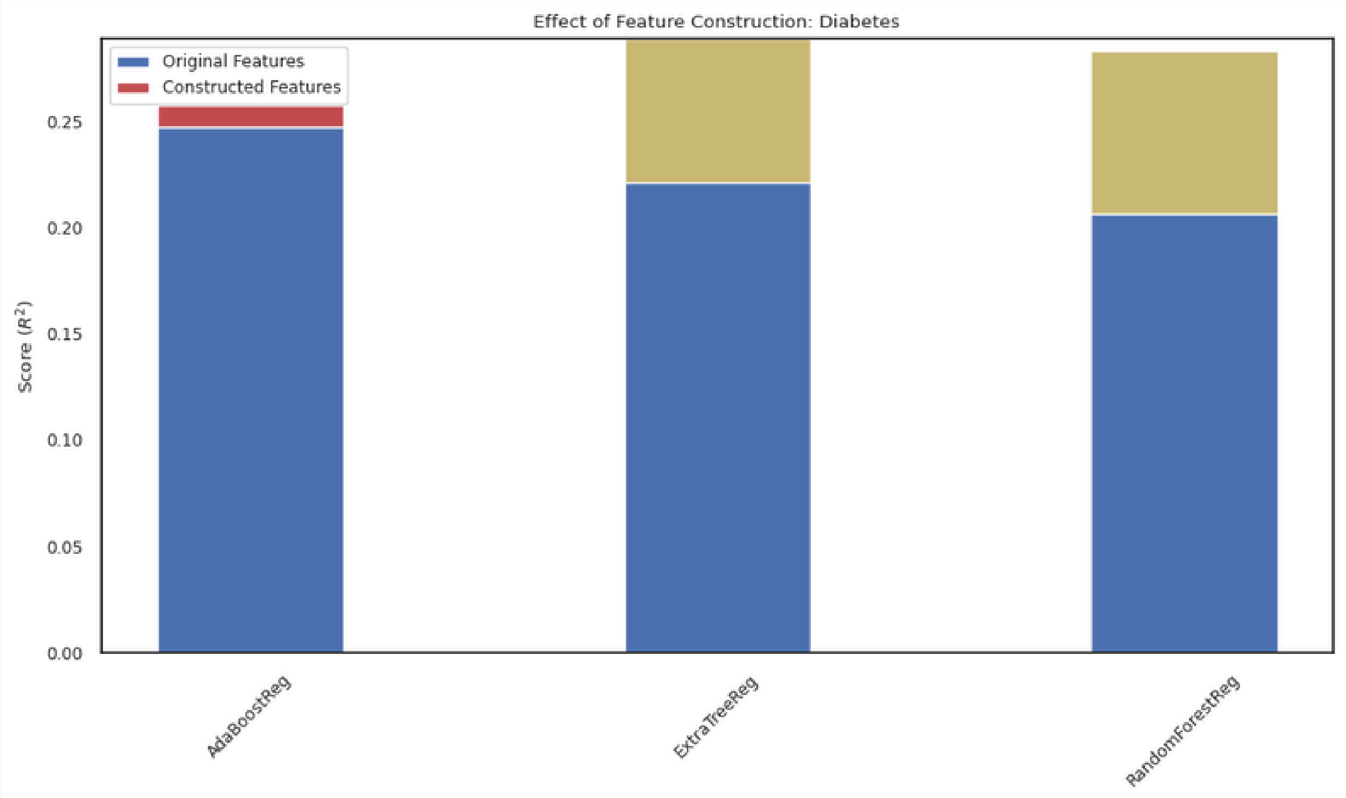
M3GP: Creating New Individual

- **crossovers:**
  - STXO:
    - randomly swaps two nodes of two individuals
    - returns the two individuals as offsprings
  - M3XO:
    - randomly swaps two features of two individuals
- **mutations:**
  - STMUT
    - randomly selects one node from a single individual
    - replaces the node with a newly generated node
  - M3ADD
    - randomly generates a new node (feature)
    - added to the list of features
  - M3REM
    - randomly removes one feature from a single individual

M3GP: Parameter Setting

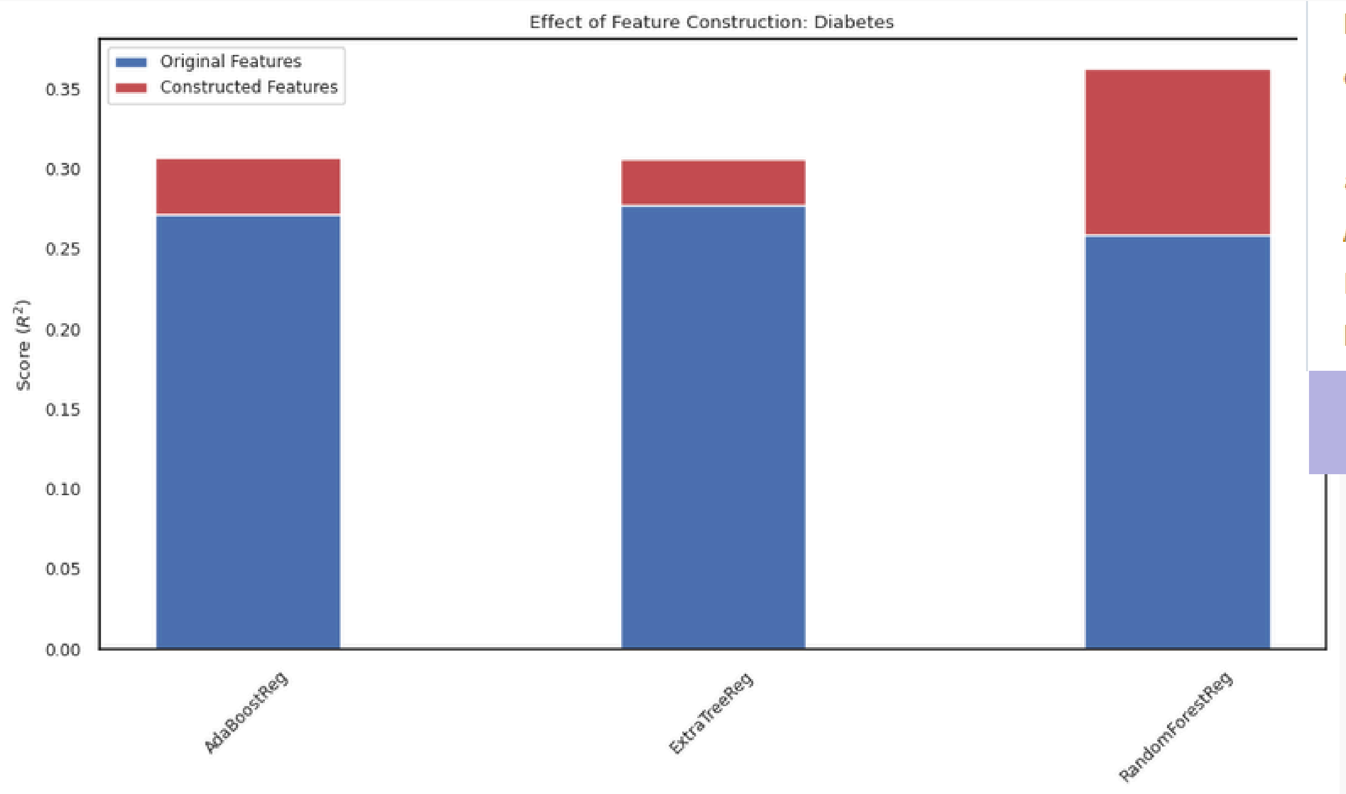
- tried many combinations
- high effect of random state (mainly for Diabetes ds)
- did not really find a satisfactory combination

M3GP: Diabetes



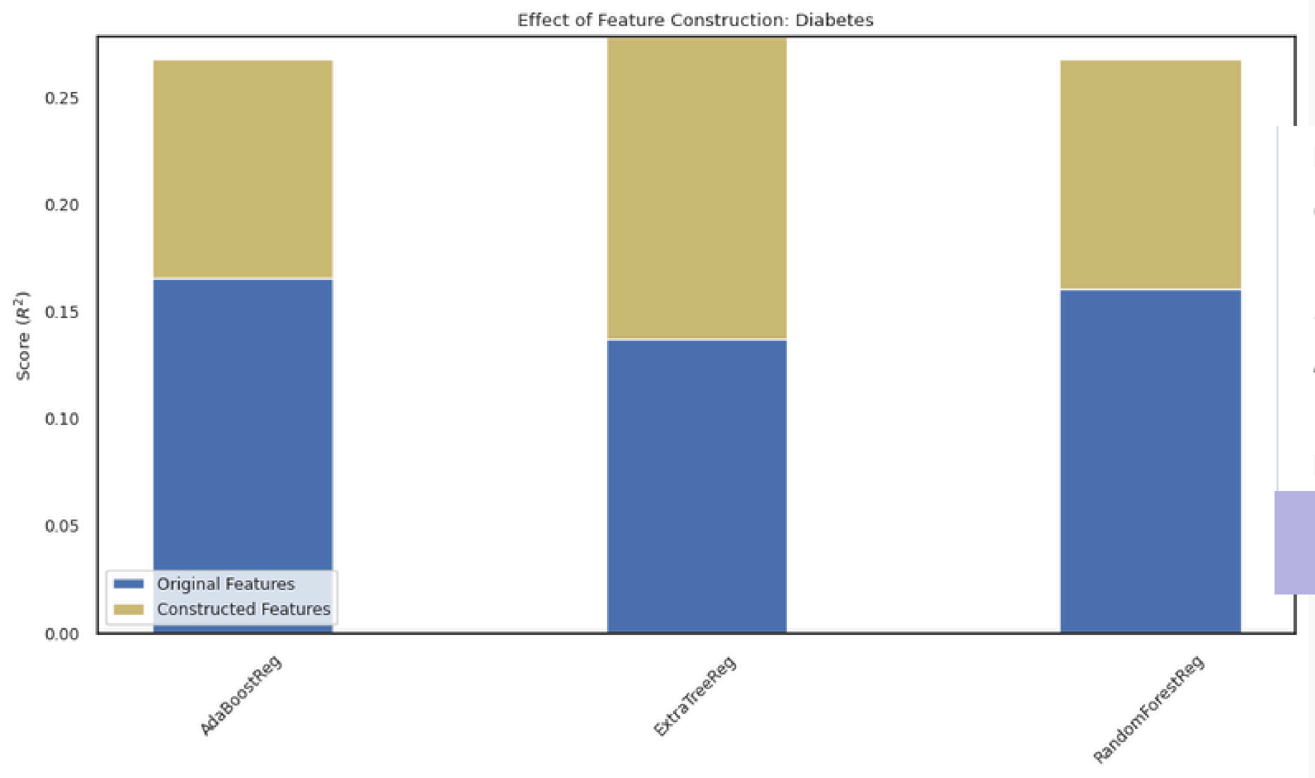

```
Results Diabetes:
orgininal number of fetures: 10, new number of features 8
                 original_score  new_score      diff
algorithm
AdaBoostReg            0.247091   0.257083  0.009991
ExtraTreeReg           0.288612   0.220617 -0.067995
RandomForestReg        0.282768   0.206057 -0.076711
```

random state: 0




```
Results Diabetes:
orgininal number of fetures: 10, new number of features 11
                 original_score  new_score      diff
algorithm
AdaBoostReg            0.270867   0.306361  0.035494
ExtraTreeReg           0.276744   0.305372  0.028628
RandomForestReg        0.258466   0.362617  0.104151
```

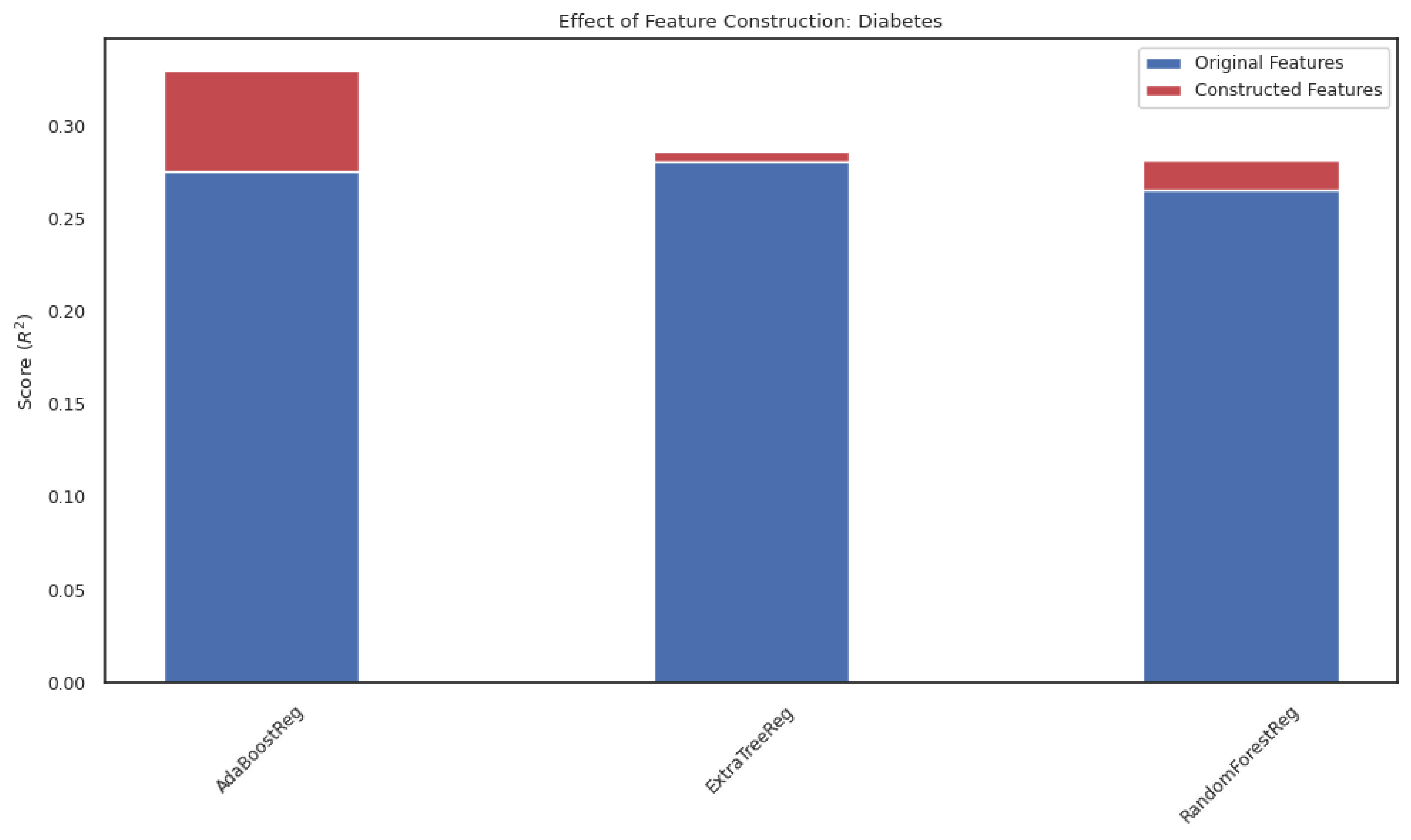
random state: 1




```
Results Diabetes:
orgininal number of fetures: 10, new number of features 9
                 original_score  new_score      diff
algorithm
AdaBoostReg            0.267802   0.165340 -0.102461
ExtraTreeReg           0.277900   0.137080 -0.140820
RandomForestReg        0.267796   0.160382 -0.107414
```

random state: 2

M3GP: Diabetes



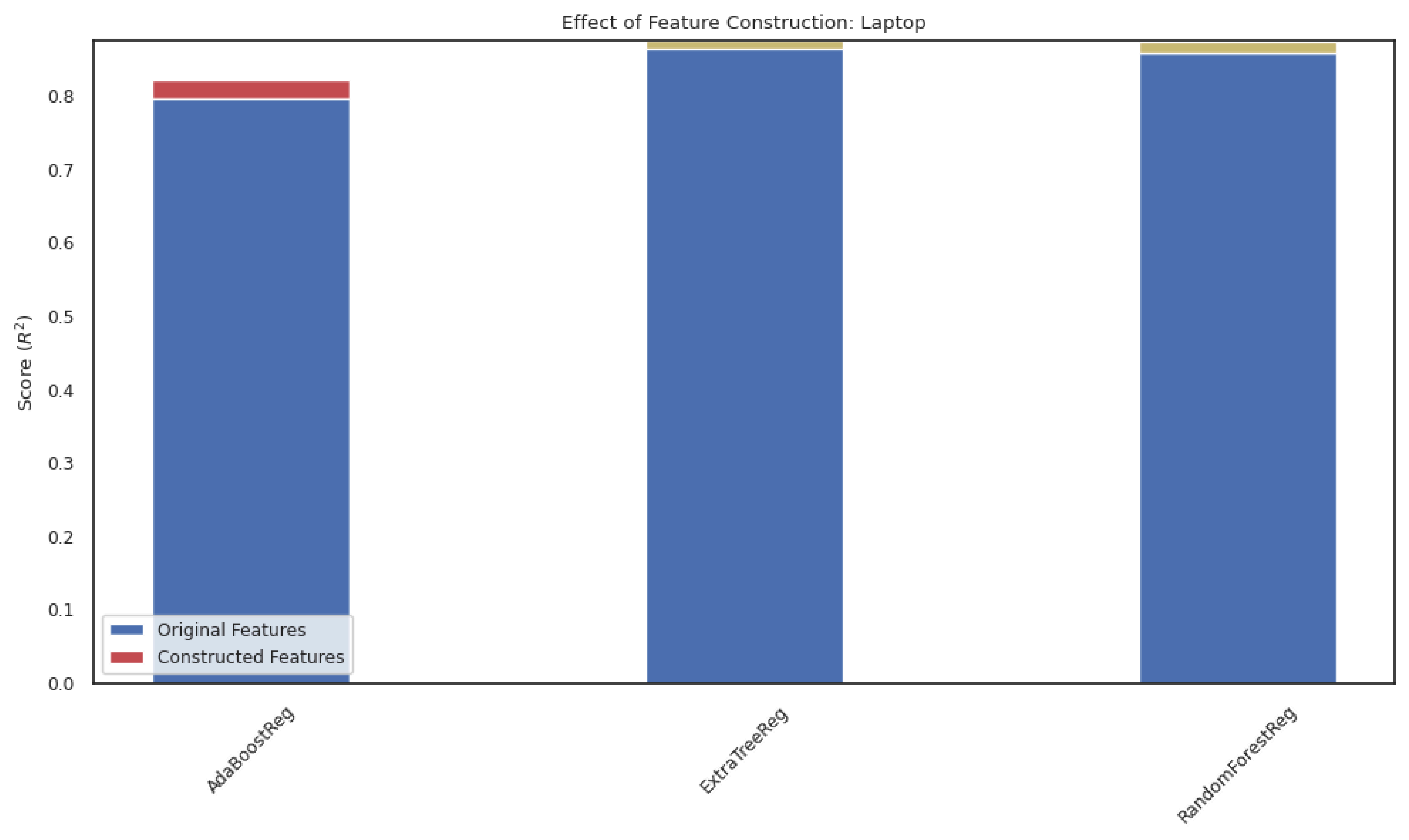
Results Diabetes:

orgininal number of fetures: 10, new number of features 9

algorithm	original_score	new_score	diff
AdaBoostReg	0.275095	0.330138	0.055043
ExtraTreeReg	0.280661	0.286292	0.005631
RandomForestReg	0.265556	0.281845	0.016289

random state: 3

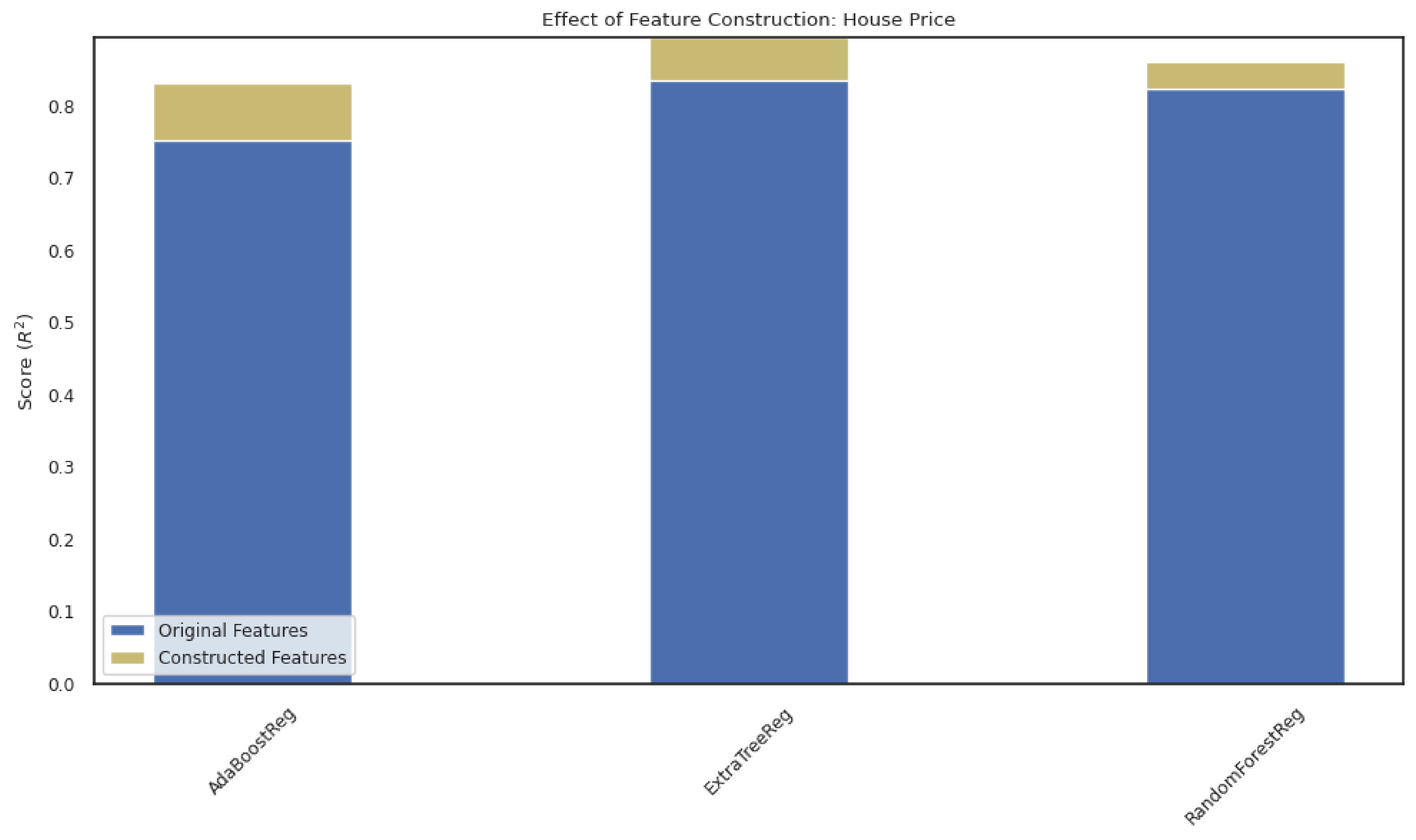
M3GP: Laptop Prices




```
Results Laptop:
orgininal number of fetures: 11, new number of features 11
                   original_score  new_score      diff
algorithm
AdaBoostReg              0.796998   0.821019  0.024021
ExtraTreeReg             0.875590   0.864771 -0.010819
RandomForestReg          0.873309   0.857835 -0.015474
```

```
> Parameters
  > Random State:       42
  > Operators:          [('abs', 1), ('pow2', 1), ('+', 2), ('-', 2), ('*', 2), ('/', 2), ('min', 2), ('max', 2)]
  > Population Size:    100
  > Max Generation:     20
  > Tournament Size:    5
  > Elitism Size:       1
  > Max Initial Depth:  5
  > Max Depth:          17
  > Minimum Dimensions: 5
  > Maximum Dimensions: 5000
  > Wrapped Model:      DecisionTreeRegressor
  > Fitness Type:       MSE
  > Threads:            1
```

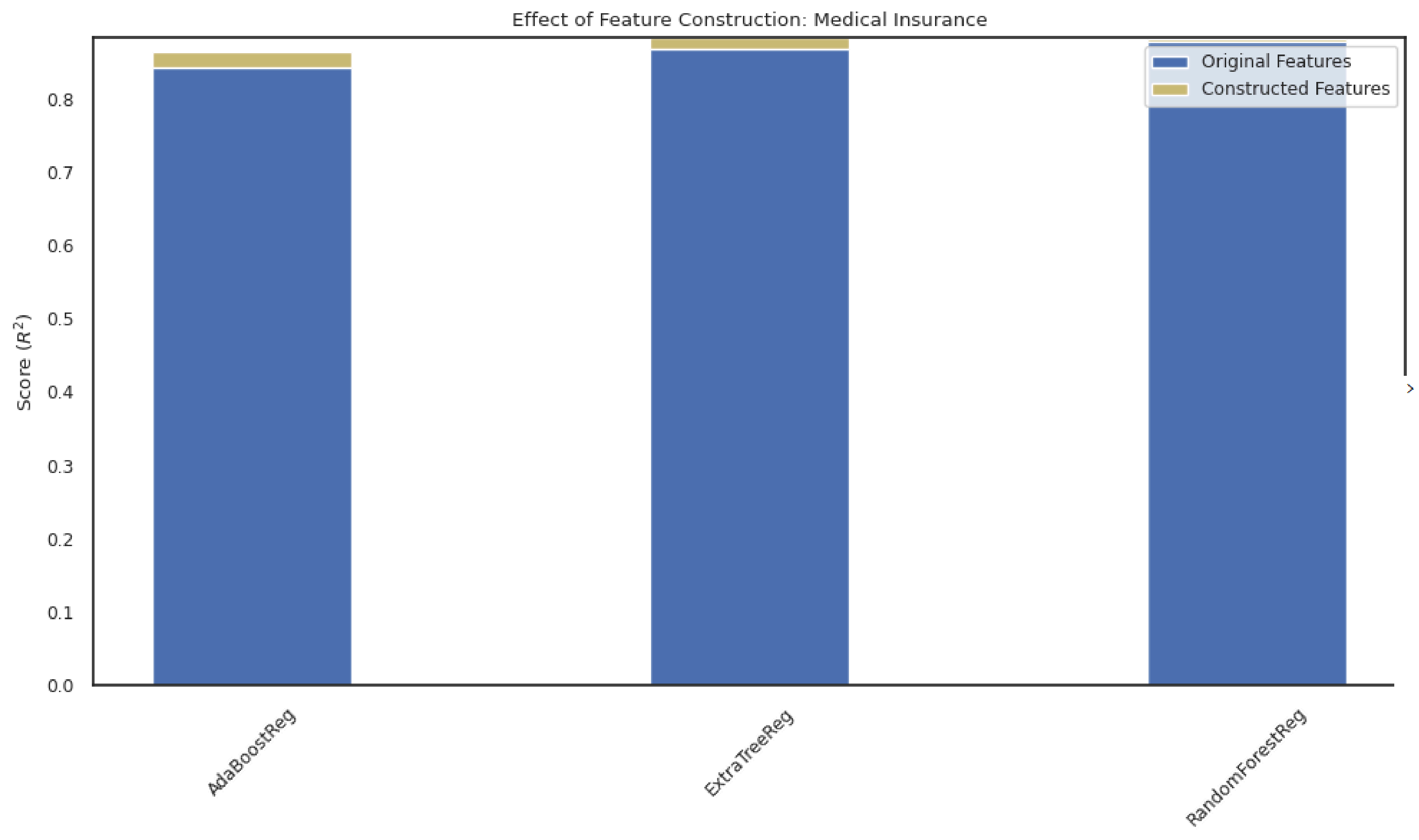
M3GP: House Prices




```
Results House Price:
orgininal number of fetures: 18, new number of features 11
                    original_score   new_score      diff
algorithm
AdaBoostReg              0.830453    0.751239 -0.079214
ExtraTreeReg             0.893806    0.834407 -0.059400
RandomForestReg          0.859627    0.823156 -0.036471
```

```
> Parameters
  > Random State:       42
  > Operators:          [('abs', 1), ('pow2', 1), ('+', 2), ('-', 2), ('*', 2), ('/', 2), ('min', 2), ('max', 2)]
  > Population Size:    100
  > Max Generation:     20
  > Tournament Size:    5
  > Elitism Size:       1
  > Max Initial Depth:  5
  > Max Depth:          17
  > Minimum Dimensions: 9
  > Maximum Dimensions: 5000
  > Wrapped Model:      DecisionTreeRegressor
  > Fitness Type:       MSE
  > Threads:            1
```

M3GP: Medical Insurance



Results Medical Insurance:

orgininal number of fetures: 6, new number of features 7

algorithm	original_score	new_score	diff
AdaBoostReg	0.864742	0.843586	-0.021156
ExtraTreeReg	0.883954	0.869317	-0.014637
RandomForestReg	0.881992	0.877448	-0.004544

> Parameters
- > Random State: 42
- > Operators: [('abs', 1), ('pow2', 1), ('+', 2), ('-', 2), ('*', 2), ('/', 2), ('min', 2), ('max', 2)]
- > Population Size: 100
- > Max Generation: 20
- > Tournament Size: 5
- > Elitism Size: 1
- > Max Initial Depth: 5
- > Max Depth: 17
- > Minimum Dimensions: 3
- > Maximum Dimensions: 5000
- > Wrapped Model: DecisionTreeRegressor
- > Fitness Type: MSE
- > Threads: 1

M3GP: Summary

- tested on four datasets
- results heavily depended on the random state
- did not find satisfactory combination of parameters
- time-consuming
- no clear improvement

Comparison: EF, M3GP

- frameworks focusing on **FC using GP**
- **individual** = set of new features, not just one feature
- use **crossover** and **mutation**
- **M3GP** - operators for adding and removing features
- **EF** - fixed number of features (set during init)

Comparison: EF, M3GP

- highly dependent on **random_state**
- obtain both reasonable and bad results
- new features - enhance and decrease the final score outcome

- **EF** - brought fairly better results
  - in terms of usability (time)

- **a bit overkill** - very **complex methods**, **not that satisfactory** results
  - maybe due to the dataset
  - it would need further investigation
  - test it on more datasets
  - hyper-parameter tuning (Optuna, Grid Search)

Sources, Articles

- **FC SOTA:**
  - Vouk, B., Guid, M., Robnik-Sikonja, M.: Feature construction using explanations of individual predictions. Engineering Applications of Artificial Intelligence 120, 105823 (2023). https://doi.org/https://doi.org/10.1016/j.engappai.2023.105823

- **GP (figure):**
  - https://www.researchgate.net/figure/Genetic-programming-Tree-based-crossover_fig4_282769665

- **Pipeline (figure):**
  - https://www.heavy.ai/technical-glossary/feature-engineering

- **Evolutionary Algorithm (figure):**
  - https://is.muni.cz/auth/el/fi/podzim2023/IV126/um/3.pdf

Sources, Articles

- **Evolutionary Forest:**
  - articles:

    - H. Zhang, A. Zhou and H. Zhang, "An Evolutionary Forest for Regression," in IEEE Transactions on Evolutionary Computation, vol. 26, no. 4, pp. 735-749, Aug. 2022, doi: 10.1109/TEVC.2021.3136667.
    - H. Zhang, A. Zhou, Q. Chen, B. Xue and M. Zhang, "SR-Forest: A Genetic Programming based Heterogeneous Ensemble Learning Method," in IEEE Transactions on Evolutionary Computation, doi: 10.1109/TEVC.2023.3243172.

  - github implementation:
    - https://github.com/hengzhe-zhang/EvolutionaryForest

Sources, Articles

- **M3GP:**
  - articles:
    - J. E. Batista and S. Silva, "Comparative study of classifier performance using automatic feature construction by M3GP," 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 2022, pp. 1-8, doi: 10.1109/CEC55065.2022.9870343.
    - Muñoz, L., Trujillo, L., & Silva, S. (2015). M3GP - multiclass classification with GP. In Genetic Programming - 18th European Conference, EuroGP 2015, Proceedings (Vol. 9025, pp. 78-91). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 9025). Springer-Verlag. https://doi.org/10.1007/978-3-319-16501-1_7
  - github implementation:
    - https://github.com/jespb/Python-M3GP