EXERCISE 2: IMPLEMENTATION

Rois Lea Michaela (11810807)
Bhardwaj Aman (12333472)
Haubenburger Gabriel (11840531)

INTRODUCTION

In this exercise, we implemented a Random Forest algorithm (based on regression

Trees) for predicting numeric values. We selected two datasets, namely, colleges (large, high dimensional with missing values) and real estate valuation (small, low dimensional and no missing value). The colleges dataset has 51 attributes which are numeric, ordinal and nominal. The target attribute is taken as “percent_pell_grant”. The real estate valuation dataset has 8 attributes which are all numeric and the target attribute is “Y house price of unit area”.

Datasets

- Real Estate Valuation
  - 6 features + target
  - 414 instances
- Colleges
  - 48 features
  - 7063 instances
  - Large number of missing values

Preprocessing:

- Real Estate Valuation:

1. Dropped the No column.
2. Transformed the X1 transaction date (which was in numeric decimal form) to Year and Month and placed these as the First and Second columns.
3. Scaled the X3 distance to the nearest MRT station column using Robust scalar as it had a large range of values. The last column “Y house price of unit area” is the target attribute.

- Colleges:

1. Dropped some columns unnecessary for the analysis like school website, latitude, longitude, school name (as school can be identified by UNITID), zip and other columns having all missing values and all same values.
2. Used Mode (Most frequent) to impute and handle the missing values of Categorical columns . Used Median for imputing missing values in numeric columns.
3. Categorical columns were encoded by Label Encoder.
4. Using Robust Scaler, applied scaling for some columns which had large difference between values or large range of values.
5. Applied Feature Selection (Variance Threshold), threshold = 0.02, to remove features having very small variance.
6. Finally, we selected 'percent_pell_grant’ as the target variable and save this preprocessed dataset to a csv file.

Decision Tree Regressor

- A variant of the simple decision tree model for regression tasks
- Parameters and Flexibility:
  - Supports configurable depth, leaf sizes, and splitting strategies
  - Offers deterministic (best) and stochastic (random) splitting
- Efficiency:
  - Computationally expensive for large datasets (random splitter reduces the cost a bit)
  - Lack of advanced optimizations (pre-pruning, post-pruning)

Random Forest Regressor

- Uses multiple decision trees trained on random subsets of the whole data set
- Counters inherent instability of decision trees
- Key features:
  - Custom or sklearn trees
  - Bootstrap sampling
  - Aggregation of predictions
  - Flexible parameters
  - Reproducibility (random_state)

Hyperparameter optimization

- We experimented with:
  - Number of trees
  - A maximum of features considered in a split
  - Maximum tree depth
  - A minimum of samples before a split
  - A minimum of samples per leaf
  - A maximum of leaf nodes
  - A maximum of samples for each tree

Comparison to scikit learn DecisionTreeRegressor

- Much faster training times
- Faster prediction times
- Way worse scores overall (overfitting)

Comparison to scikit learn DecisionTreeRegressor

- Real estate valuation: clf = RandomForestRegressor(use_skl_tree=False, max_samples=100, max_features=None)
- # clf = DTRegressor(splitter='random')

	Training time [s]	Prediction Time [s]	R^2 score	MSE	MAE
Real estate DT	0.222	0.000	0.346	111.324	6.043
Real estate from scratch	7.494	0.017	0.691	52.494	4.728

Comparison to scikit learn DecisionTreeRegressor

- Colleges: clf = DTRegressor(splitter='random', max_leaf_nodes=100, verbose=True, random_state=random_state)
- # clf = SKDT(splitter='random', random_state=random_state)

	Training time [s]	**Prediction Time [s]**	**R² score**	**MSE**	**MAE**
Colleges Scikit learn DT	0.0227	0.001	0.066	0.0469	0.152
Colleges DT from scratch	8.700	0.000	0.357	0.032	0.1337

Comparison to scikit-learn RandomForestRegressor

- Overall much faster training times
- Slightly smaller prediction times
- Roughly comparable, but usually slightly better scores

Comparison to scikit-learn RandomForestRegressor

- Real estate valuation: clf = RandomForestRegressor(use_skl_tree=True)
- clf = RandomForestRegressor(use_skl_tree=False, max_samples=100, max_features=None)

	Training time [s]	**Prediction Time [s]**	**R^2 score**	**MSE**	**MAE**
Real estate Scikit-learn	0.062	0.007	0.636	61.982	5.054
Real estate from scratch	7.494	0.017	0.691	52.494	4.728

Comparison to scikit-learn RandomForestRegressor

```
clf = RandomForestRegressor(use_skl_tree=True, max_samples=500, max_features=30,random_state=random_state, n_estimators=50)
clf = RandomForestRegressor(use_skl_tree=False, max_samples=500, max_features=30,random_state=random_state, n_estimators=50)
```

	Training time [s]	Prediction Time [s]	R^2 score	MSE	MAE
Colleges Scikit-learn	0.159	0.0119	0.495	0.025	0.115
Colleges from scratch	105.571	0.213	0.4749	0.026	0.118

Comparison to LLM version

- Much shorter code (less parameters)
- Seems to also work on the preprocessed data sets

- LLM:
  - Uses pre-implemented class -- > Less complex and less coding effort
  - Less focus on details in splitting criteria
  - Benefits from optimizations in libraries (like sklearn – uses highly efficient compiled code
- Scratch: requires deep knowledge of algorithmic details
  - More bugs possible
  - Performance may lag

Comparison to LLM version

- Real estate valuation:

	Training time [s]	Prediction Time [s]	R^2 score	MSE	MAE
Real estate LLM	0.849	0.001	0.465	91.233	7.043
Real estate from scratch	5.547	0.021	0.683	53.990	4.808

- Insights:
  - LLM: faster training and prediction times, but sacrifies accuracy → lower R^2, MSE, and MAE.
  - RF from scratch: slower training, but better accuracy, lower MSE and MAE

Comparison to LLM version

- Colleges:

	Training time [s]	Prediction Time [s]	R^2 score	MSE	MAE
Colleges LLM	100.707	0.033	0.012	0.050	0.187
Colleges from scratch	21.159	0.376	0.431	0.029	0.127

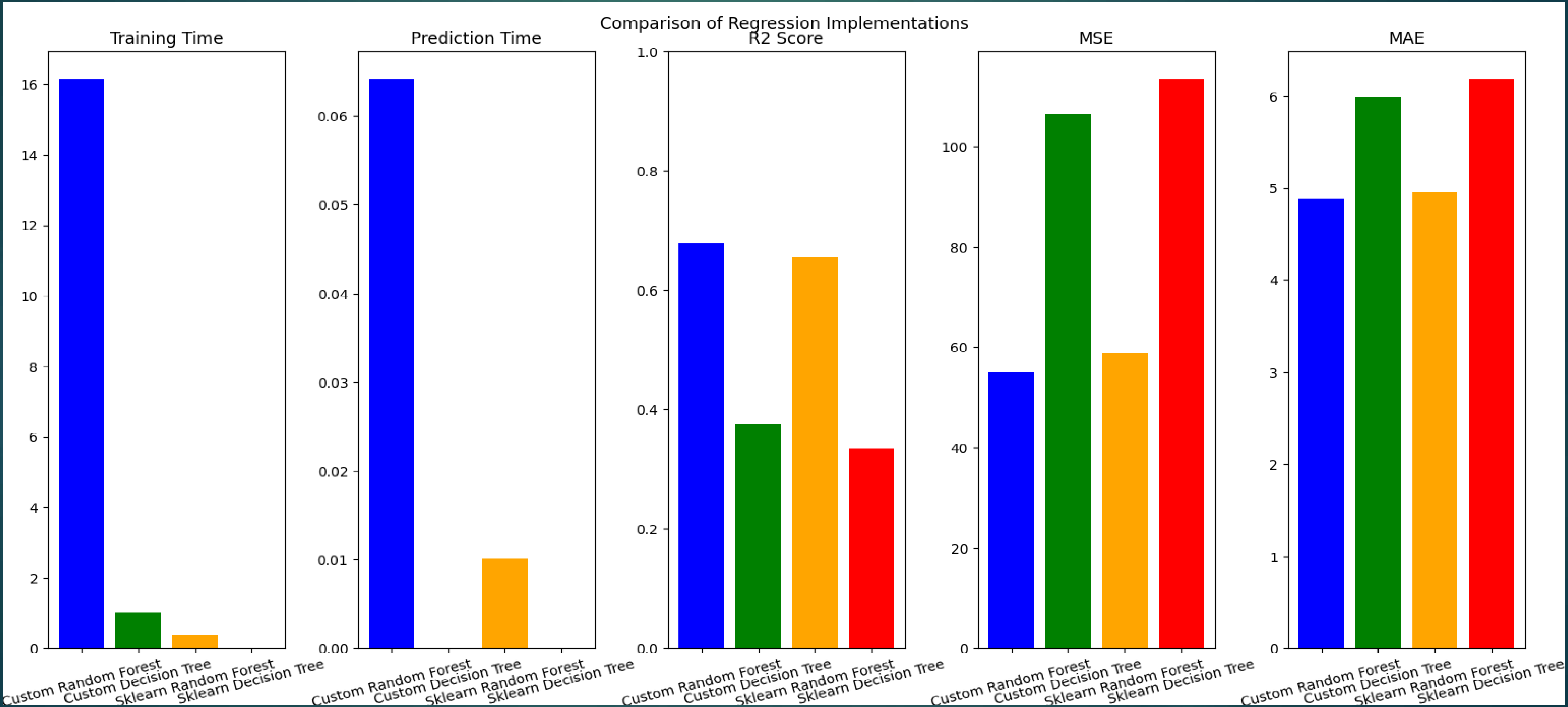
- Insights:
  - LLM: Slower training time, faster prediction time. But poor model performance → very low R^2 and higher error metrics compared to RF from scratch
  - RF from scratch: 5 x faster training times, but lower prediction times compared to LLM model. Better accuracy with higher R^2, lower MSE and MAE.

Efficiency

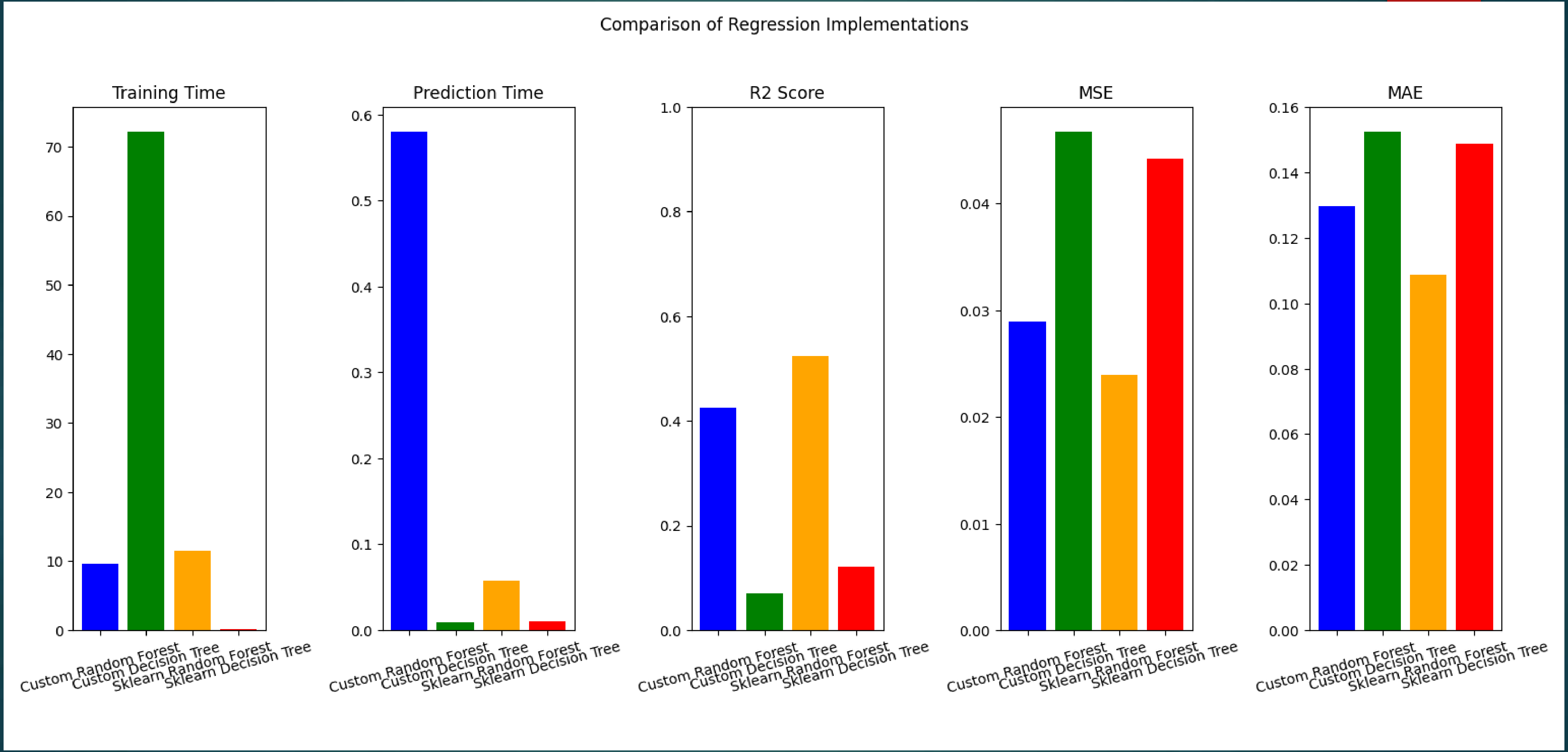
- Our implementation had a notable gap in efficiency compared to the scikit learn version
- Mainly from using own trees
- Possible reasons:
  - Penalty from implementation in an interpreted language
  - Lacking optimization
  - Bugs

Plots:

Real Estate



Colleges:


Comparison of Regression Implementations
Training Time
Prediction Time
R2 Score
MSE
MAE
Custom Random Forest
Custom Decision Tree
Sklearn Random Forest
Sklearn Decision Tree