COCS 5557: Practical Machine Learning

Pipeline Optimization

Introduction:

The objective of this study is to optimize the entire machine learning pipeline, including preprocessing steps and algorithm selection, through *hyperparameter optimization*. We aim to explore the efficiency of various components within the pipeline and their interactions. By employing hyperparameter optimization techniques, we seek to identify the most effective combination of preprocessing steps, algorithms, and hyperparameters for each dataset.

Key Concepts Deployed:

- **Optuna**: Optuna is a Python library for hyperparameter optimization. It suggests hyperparameters based on the distributions specified in the objective function and uses these results to suggest better hyperparameters in the next trials.
- **Objective Function**: This function guides the optimization process. It takes a trial object from Optuna, which it uses to suggest hyperparameters. It then trains and evaluates a model with these hyperparameters, and returns the model's accuracy.
- **Study**: The study is the main optimization process. It runs the objective function for a specified number of trials, each time with different hyperparameters suggested by Optuna. The study keeps track of the best trial - the one where the objective function returned the highest accuracy.
- **Plots**: The plots provide a visual representation of the optimization process.

Thus, the following lines of code use `Optuna` to perform `hyperparameter` optimization on a machine learning pipeline that includes `scaling, feature selection,` and select ML models.

To reiterate, the goal is to find the `hyperparameters` that `maximize` the models' accuracy on the test data. The results of the optimization are then visualized in a plot. The code is well-structured and follows the typical steps of a machine learning project: *data loading and preprocessing, model training and evaluation, hyperparameter optimization, and results visualizatio*n. The use of a pipeline and an

objective function makes the code modular and easy to modify or extend. For example, one could easily add more preprocessing steps, change the classifier/regressor, or modify the `hyperparameter` distributions. The use of Optuna makes the hyperparameter optimization process efficient and easy to manage. The plot provides a clear visualization of the results and helps to understand the effects of the `hyperparameters` on the models' performance.

The following outline the steps are specific to this exercise on the pipeline optimization

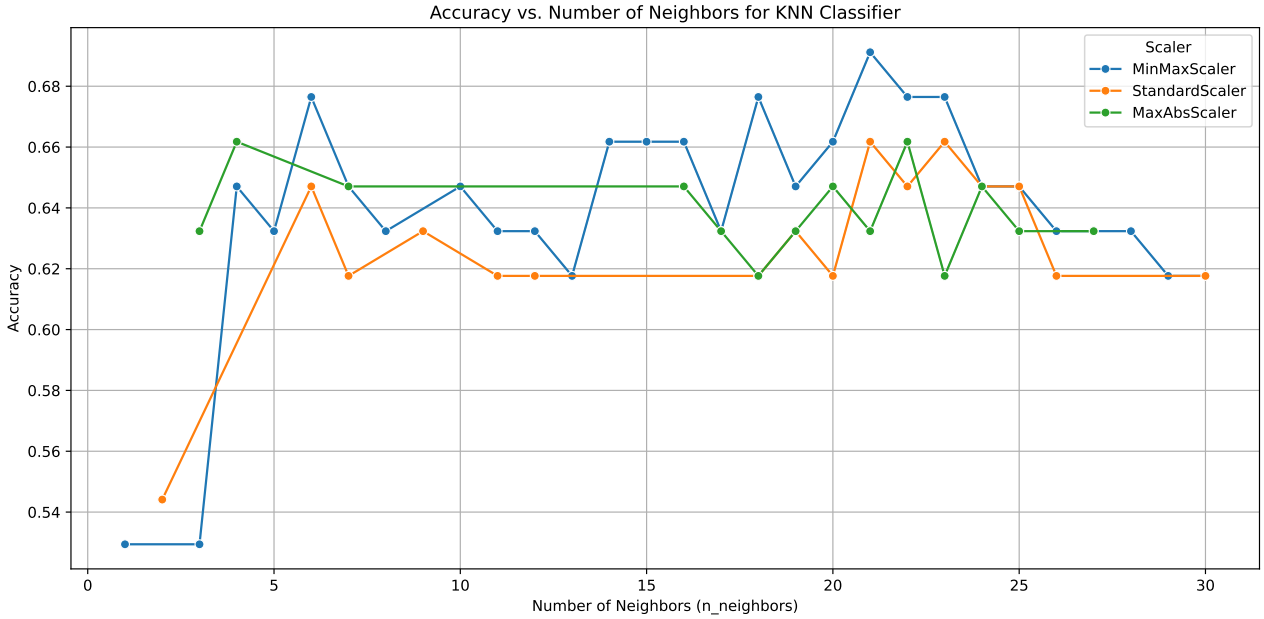
1. **Data Loading and Preprocessing**: All needed libraries and the dataset from a CSV file were loaded. The dataset is then split into features (`X`) and the target variable (`y`). For the classification part, the target variable is encoded into unique integers using `LabelEncoder`. The feature and target datasets were individually split into training and testing sets using `train_test_split` from `sklearn`.
2. **Objective Function**: The objective function uses a trial object from `Optuna` to suggest `hyperparameters` for the model being considered. The function suggests a `scaler (StandardScaler, MinMaxScaler, or MaxAbsScaler)`, a `threshold` for the `VarianceThreshold` selector, and the `number of neighbors` for the `KNeighborsClassifier`, for instance. A pipeline is then created with the chosen `scaler, selector`, and `classifier`. *The pipeline is fit on the* `training data` *and evaluated on the* `test data`. The accuracy of the model on the `test data` is returned as *the objective to be maximized*.
3. **Optimization**: Following the immediate step, an `Optuna study` is created to `maximize` the `objective function`. The `study` runs the `objective function` for 700 and 100 trials for the classification and regression parts respectively. Each time, the `study` runs with different `hyperparameters` suggested by `Optuna`. The study keeps track of the performance of each trial. The best `trial` and its `accuracy` and `hyperparameters` are then returned.
4. **Results Conversion and Plotting**: `study.trials_dataframe` function is then used to convert the results of the `study` into a data frame for easy manipulation and plotting. The data frame is sorted, for instance, by the number of neighbors, and a line plot is created to visualize the accuracy against the `number of neighbors` for the `KNN classifier`. different lines represent different `scalers`.

Classification Part

KNN classifier

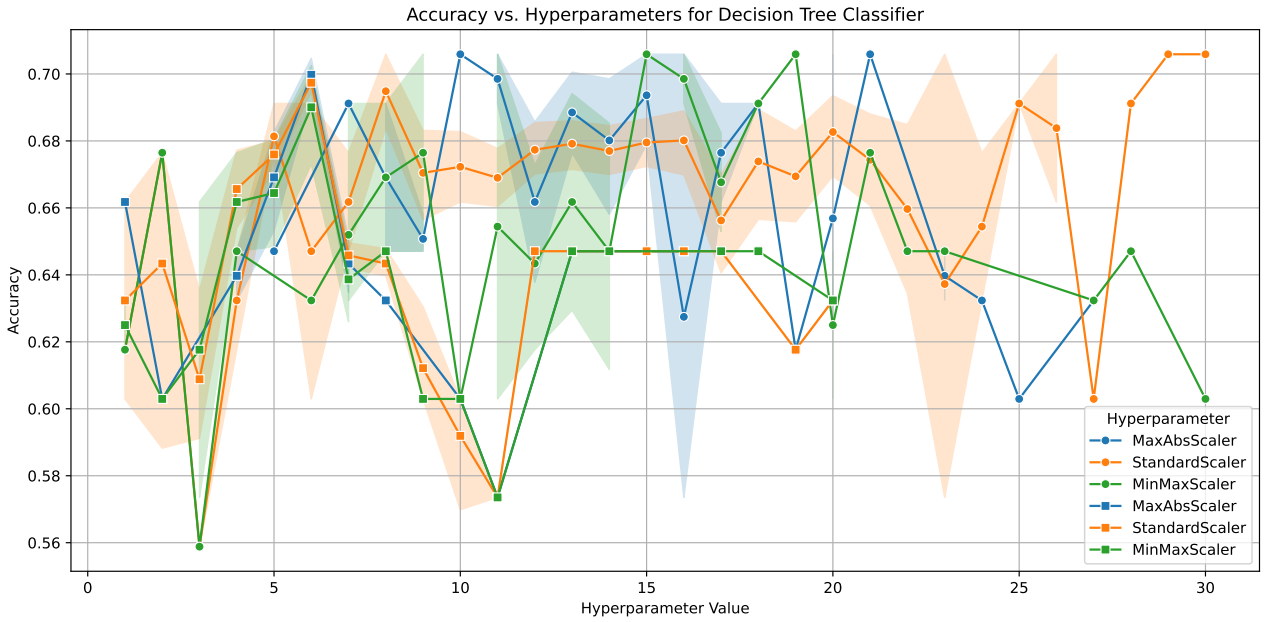
```
Accuracy w/o Pipeline Optimization: 0.6764705882352942
Best trial:
Accuracy: 0.6911764705882353
Best hyperparameters:
{'scaler': 'MinMaxScaler', 'selector__threshold': 0.0059814187301486,
```

```
'classifier__n_neighbors': 21}
Index(['number', 'value', 'datetime_start', 'datetime_complete', 'duration',
       'params_classifier__n_neighbors', 'params_scaler',
       'params_selector__threshold', 'state'],
      dtype='object')
```



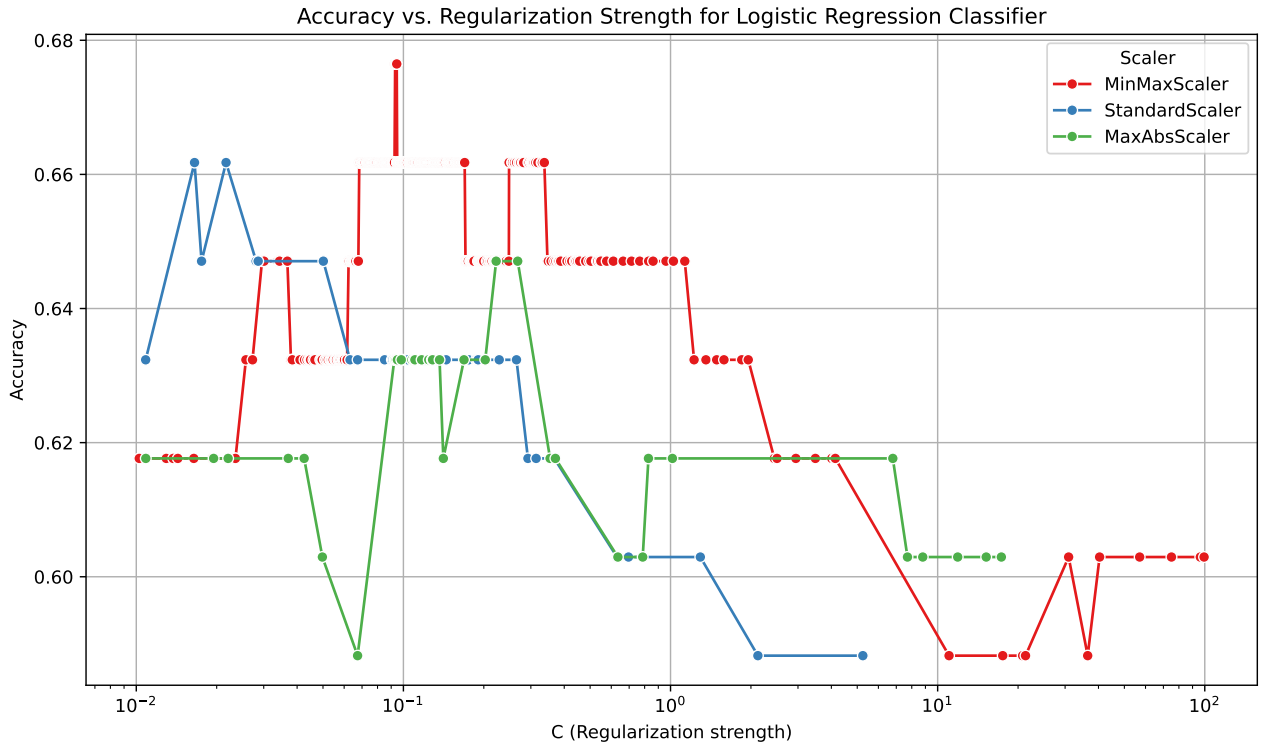
Decision tree classifier

```
Accuracy w/o Pipeline Optimization: 0.5588235294117647
Best trial:
Accuracy: 0.7058823529411765
Best hyperparameters:
{'scaler': 'StandardScaler', 'selector__threshold': 0.00767968748815254,
'classifier__max_depth': 20, 'classifier__min_samples_split': 16,
'classifier__min_samples_leaf': 6}
Index(['number', 'value', 'datetime_start', 'datetime_complete', 'duration',
       'params_classifier__max_depth', 'params_classifier__min_samples_leaf',
       'params_classifier__min_samples_split', 'params_scaler',
       'params_selector__threshold', 'state'],
      dtype='object')
```



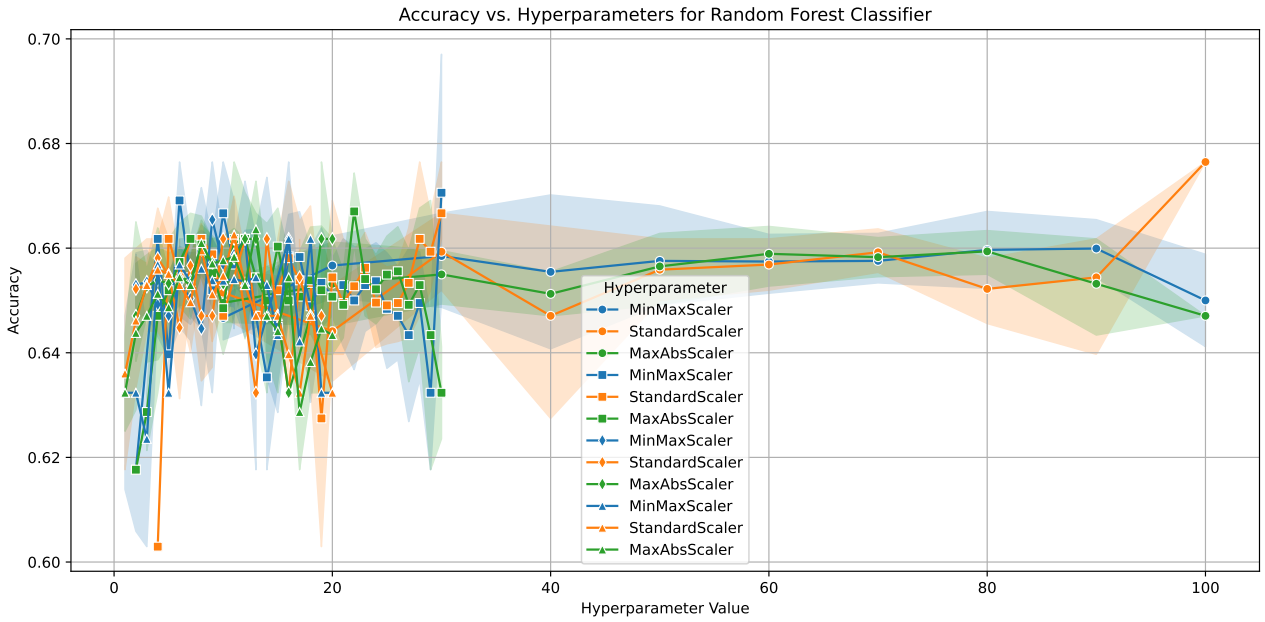
Logistic regression

```
Accuracy w/o Pipeline Optimization: 0.6470588235294118
Best trial:
Accuracy: 0.6764705882352942
Best hyperparameters:
{'scaler': 'MinMaxScaler', 'selector__threshold': 0.0004107675852508494,
'classifier__C': 0.09313330261328194}
Index(['number', 'value', 'datetime_start', 'datetime_complete', 'duration',
       'params_classifier__C', 'params_scaler', 'params_selector__threshold',
       'state'],
      dtype='object')
```



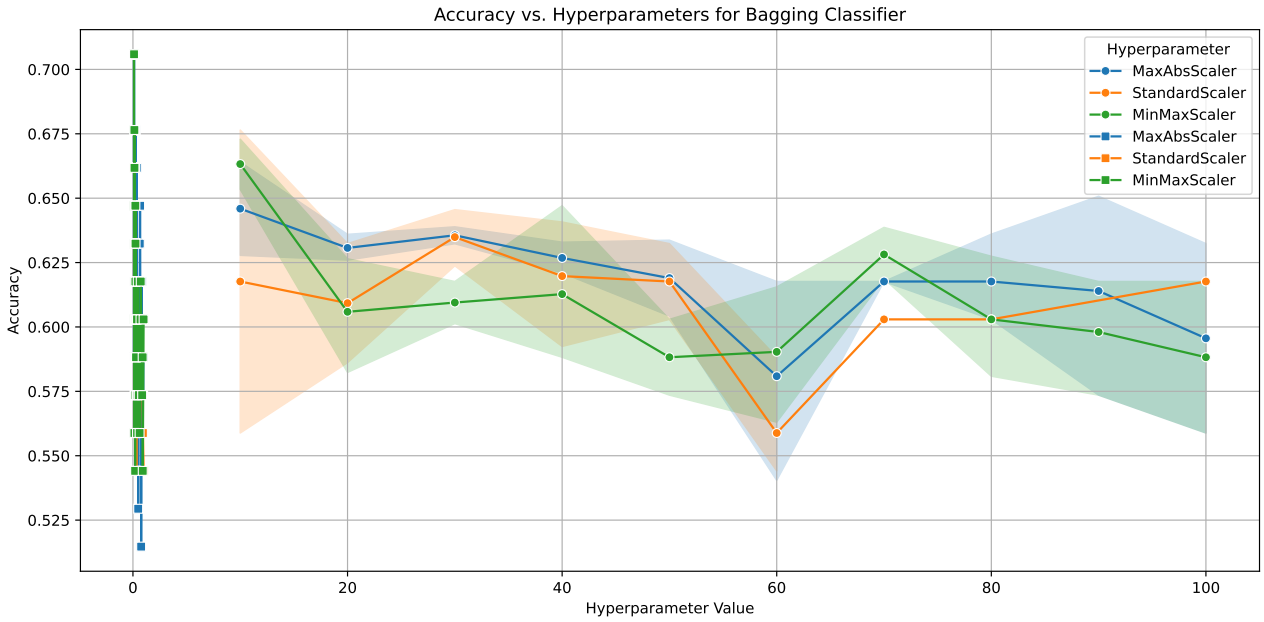
Random Forest classifier

```
Accuracy w/o Pipeline Optimization: 0.6470588235294118
Best trial:
Accuracy: 0.7205882352941176
Best hyperparameters:
{'scaler': 'MinMaxScaler', 'selector__threshold': 0.0031236420653547306,
'classifier__n_estimators': 10, 'classifier__max_depth': 30,
'classifier__min_samples_split': 2, 'classifier__min_samples_leaf': 9}
Index(['number', 'value', 'datetime_start', 'datetime_complete', 'duration',
       'params_classifier__max_depth', 'params_classifier__min_samples_leaf',
       'params_classifier__min_samples_split',
       'params_classifier__n_estimators', 'params_scaler',
       'params_selector__threshold', 'state'],
      dtype='object')
```



Bagging classifier

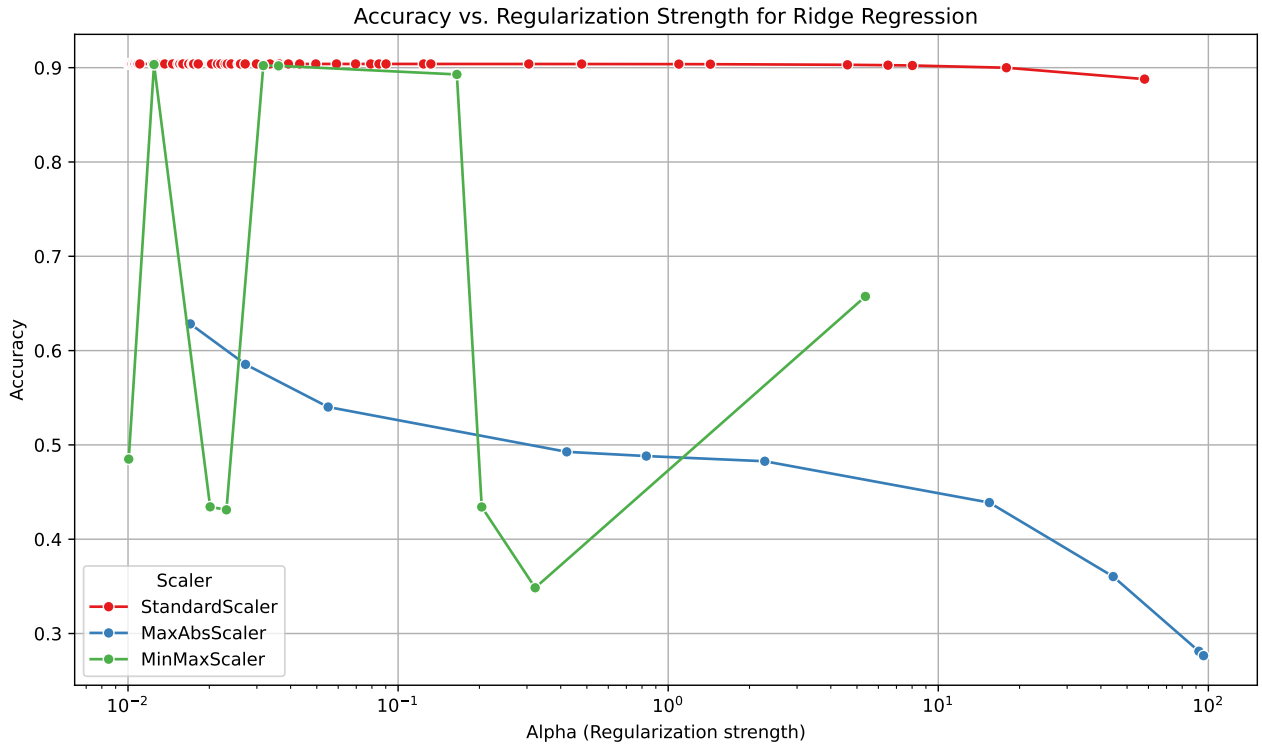
```
Accuracy w/o Pipeline Optimization: 0.6176470588235294
Best trial:
Accuracy: 0.7058823529411765
Best hyperparameters:
{'scaler': 'MaxAbsScaler', 'classifier__n_estimators': 10,
'classifier__max_samples': 0.10051415133195013}
Index(['number', 'value', 'datetime_start', 'datetime_complete', 'duration',
       'params_classifier__max_samples', 'params_classifier__n_estimators',
       'params_scaler', 'params_selector__threshold', 'state'],
      dtype='object')
```



Regression Part:

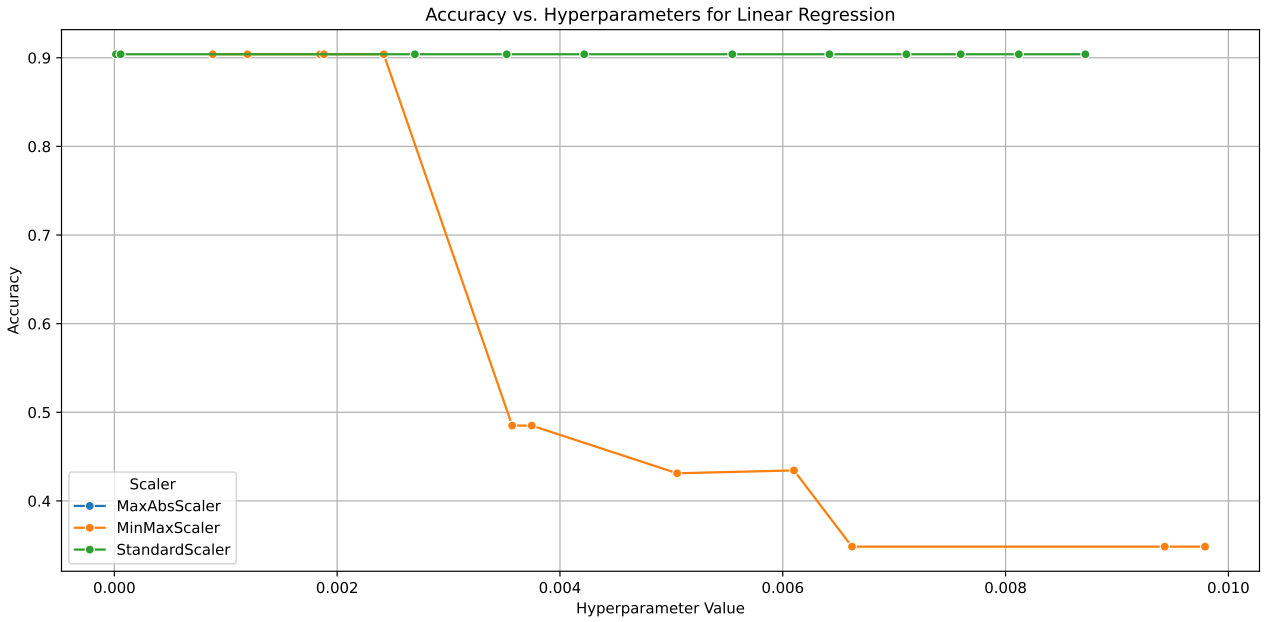
Ridge Algorithm

```
Accuracy w/o Pipeline Optimization: 0.4796389916276238
Best trial:
Accuracy: 0.9039579668873036
Best hyperparameters:
{'scaler': 'StandardScaler', 'selector__threshold': 0.006342134865264132,
'classifier__alpha': 0.010034386695118745}
```



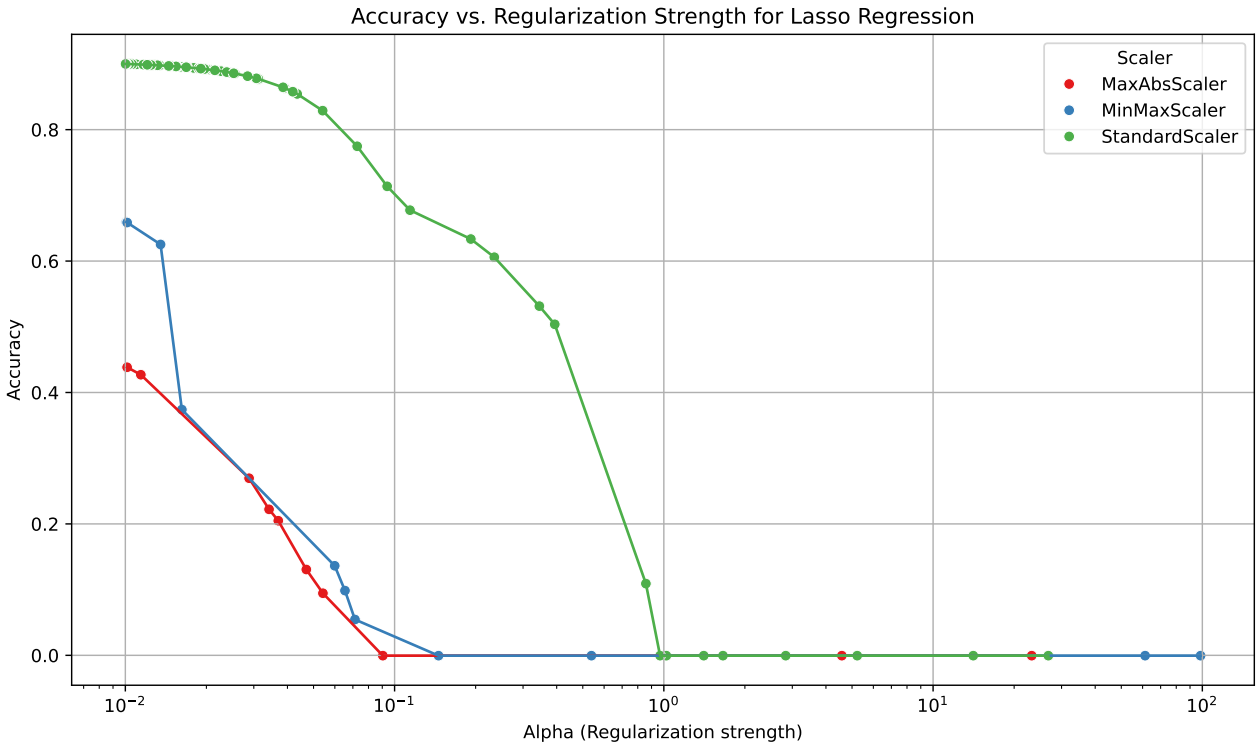
Linear Regression

```
Accuracy w/o Pipeline Optimization: 0.9039598560643549
Best trial:
Accuracy: 0.90395985606436
Best hyperparameters:
{'scaler': 'MaxAbsScaler'}
```



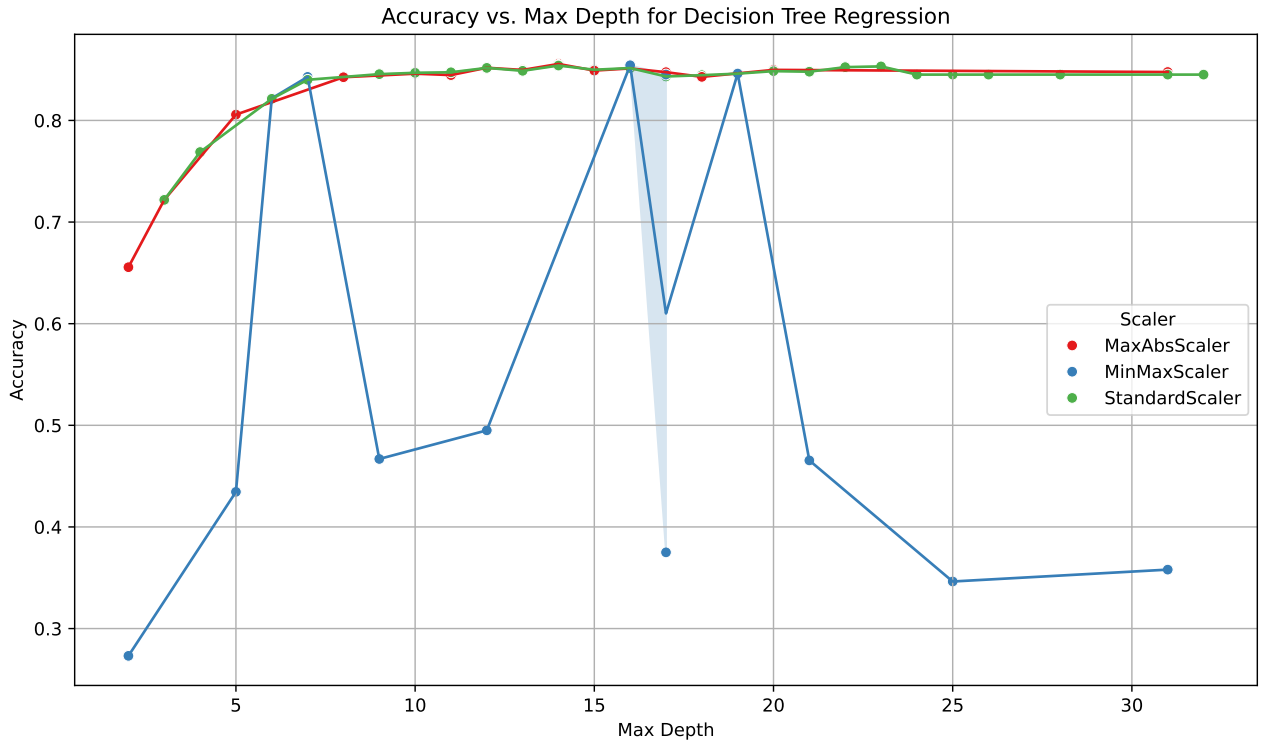
Lasso Model

```
Accuracy w/o Pipeline Optimization: 0.24001194520376679
Best trial:
Accuracy: 0.8999835936351264
Best hyperparameters:
{'scaler': 'StandardScaler', 'selector__threshold': 0.0031517564302534665,
'regressor__alpha': 0.010007240149562939}
```



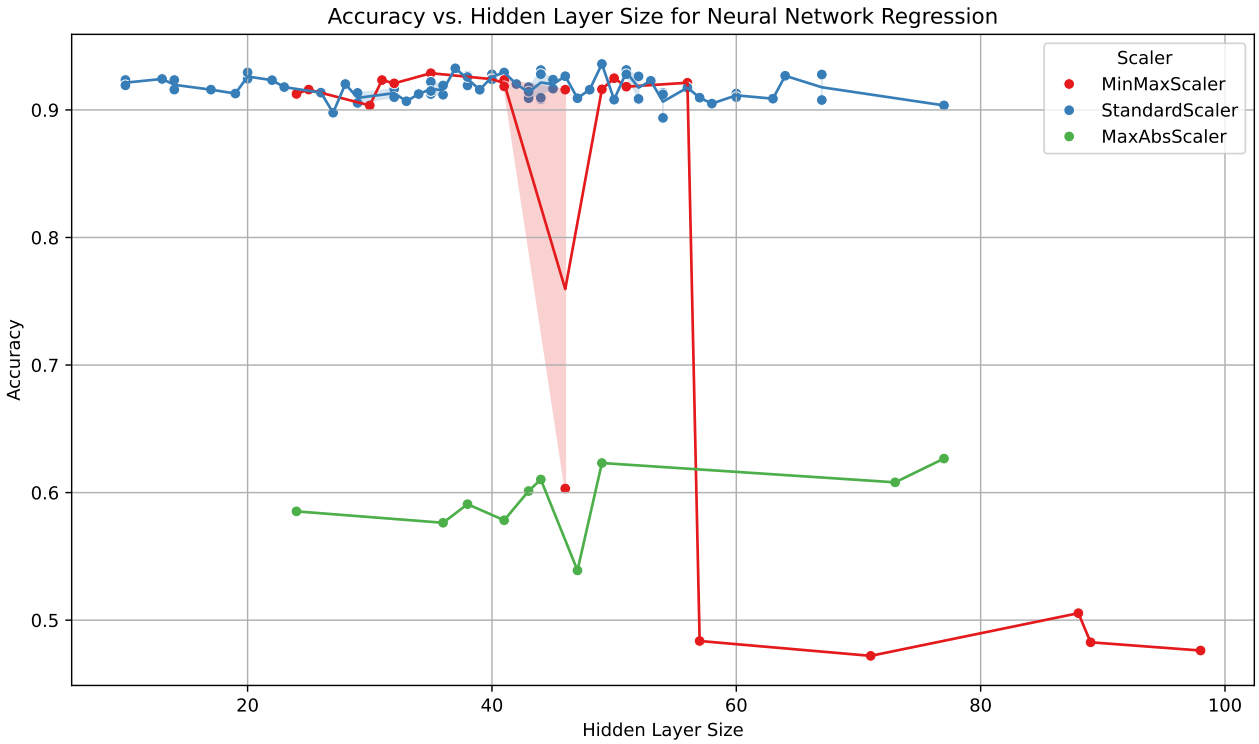
Decision Tree Regressor

```
Accuracy w/o Pipeline Optimization: 0.8472768284730636
Best trial:
Accuracy: 0.8555794556948964
Best hyperparameters:
{'scaler': 'MaxAbsScaler', 'regressor__max_depth': 14}
```



Neural Network

```
Accuracy w/o Pipeline Optimization: 0.4793176270455942
Best trial:
Accuracy: 0.9360441740744966
Best hyperparameters:
{'scaler': 'StandardScaler', 'selector__threshold': 0.002056718809844922,
'regressor__n_layers': 3, 'regressor__hidden_layer_sizes_0': 49,
'regressor__hidden_layer_sizes_1': 37, 'regressor__hidden_layer_sizes_2': 62}
```


Accuracy vs. Hidden Layer Size for Neural Network Regression
Scaler
MinMaxScaler
StandardScaler
MaxAbsScaler
Accuracy
0.9
0.8
0.7
0.6
0.5
20
40
60
80
100
Hidden Layer Size