COSC 5010-03 Practical Machine Learning Fall 2023 Pipeline Optimization Report

Michael Elgin

November 22, 2023

1 Introduction

Parts of a machine learning pipeline include not only the choice of model (algorithm) and the task of tuning the associated hyperparameters, but various choices about how to preprocess the data. This report explores the effect of normalization, feature reduction, and sample reduction (outlier removal) on the wine quality dataset[1]. The effect of these preprocessing changes is examined on a linear regression model for regression, and a decision tree and random forest for classification.

2 Dataset Description

The wine quality dataset is standard tabular data. There are 2 datasets for each color of red and white. Both contain 11 features, all of which are continuous. The target is a discrete value which is the assigned quality of the wine. For the regression tasks, the white wine dataset is used to evaluate models. For classification tasks, both datasets are concatenated, with the target being changed to be the color of the wine, with 0 being assigned to red and 1 to white.

3 Experimental Setup

All computation is done with the Python programming language. Scikit-learn is used to construct models. Pandas is used to load and preprocess the data.

For regression, performance is the mean absolute percentage error. This is defined by taking the difference between the predicted value of the model and the actual value, then dividing by the actual value. Then the mean of all of those is taken. More formally:

$$GE_{Regression\ model}(\hat{f}, \boldsymbol{X}_{test}, \boldsymbol{y}_{test}) = \frac{100}{|\boldsymbol{X}_{test}|} \sum_{i=1}^{|\boldsymbol{X}_{test}|} \left| \frac{\hat{f}(\boldsymbol{X}_{test,i}) - \boldsymbol{y}_{test,i}}{\boldsymbol{y}_{test,i}} \right|$$

[1]https://archive.ics.uci.edu/dataset/186/wine+quality

For classification, the performance metric is standard accuracy:

$$GE_{Classification\ model}(\hat{f}, \boldsymbol{X}_{test}, \boldsymbol{y}_{test}) = \frac{\sum_{i=1}^{|\boldsymbol{X}_{test}|} l_{0,1}(\hat{f}(\boldsymbol{X}_{test,i}), \boldsymbol{y}_{test,i})}{|\boldsymbol{X}_{test}|}$$

$$Acc_{Classification\ model} = (1 - GE_{Classification\ model}) * 100$$

For the linear model, no hyperparameters are tuned. For the tree and forest, Bayesian optimization is used for trying hyperparameter configurations, with a budget of 20 and 100 iterations. This tunes the max depth and minimum samples for a split in the models. The minimum max depth is 1 and minimum minimum samples for a split is 2. Their maxes are 64 and 16,384 respectively. The forest contains 100 trees.

$$\text{max depth} = 2^x, x \in [0, 6]$$

$$\text{min samples} = 2^x, x \in [1, 14]$$

The process uses nested resampling. The outer loop (for the unbiased performance estimate) uses 2 folds, the inner loop (for the search itself) uses 2 folds.

The first pipeline optimization option is whether or not to normalize the data. This consists of z-scoring the data according to the formula

$$z = \frac{(X - \mu)}{\sigma}$$

Regression will normalize the target (quality) as well, but classification will not do this to color.

The second pipeline optimization option is to reduce the amount of features in the data. This step will remove all features except for the top features that are most strongly positively or negatively correlated with the target.

The third pipeline optimization option is to reduce the amount of samples in the data by removing outliers. Outliers are determined by whether or not they are outside of the range

$$[Q1 - 1.5 \cdot IQR, Q3 + 1.5 \cdot IQR]$$

Where Q1 is the first quartile, Q3 is the third quartile, and IQR is the interquartile range. For a sample, if any value in any of the feature columns is out of range, the sample will be removed. The target is not a column considered.

For both regression and classification, the settings for the pipeline are optimized together (not independently) along with the hyperparameters as applicable.

4 Results

Table 1: Best and default pipelines explored, and their performance

Pipeline Version	Model	Normalize	Feature Amount	Sample Reduction	Max Depth	Min Samples Split	Acc/MAPE
Default	Linear	False	11	False	N/A	N/A	11.566
Optimal	Linear	False	9	False	N/A	N/A	11.564
Default	Tree	False	11	False	52	2	0.983
Optimal-20	Tree	True	7	True	34	950	0.986
Optimal-100	Tree	True	10	True	64	2	0.988
Default	Forest	False	11	False	61	2	0.995
Optimal-20	Forest	False	10	True	64	662	0.995
Optimal-100	Forest	False	9	True	64	2	0.995

pipe.ipynb prints every combination of pipeline settings with MAPE/accuracy.

4.1 Regression

The most interesting trends are plotted in figure 1.

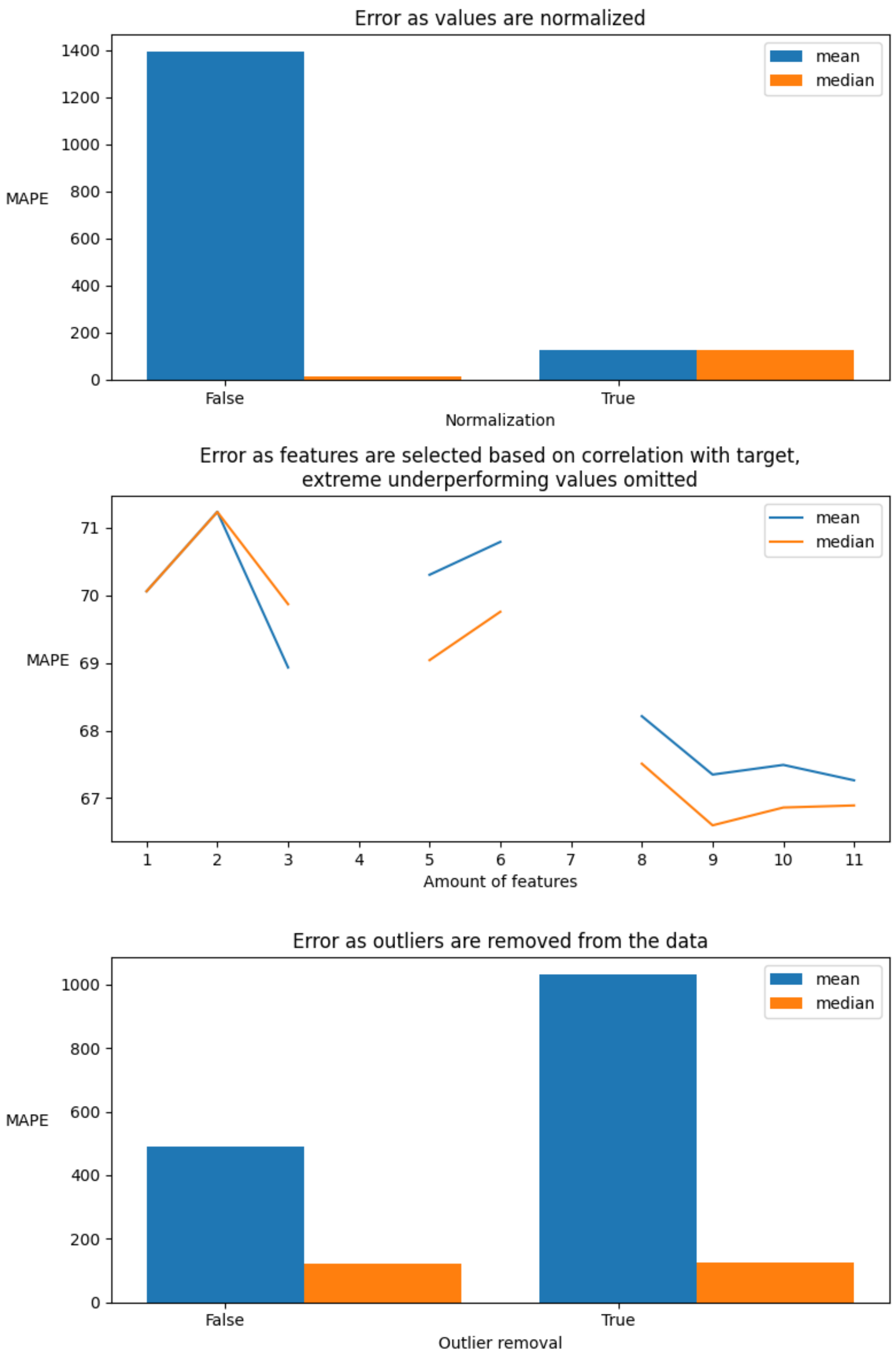

Figure 1: Regression performance trends

The individual best case was to not normalize the data, not reduce the samples, but reduce the features from 11 down to 9. This trivially changes MAPE from 11.566 in the default pipeline to 11.564. Furthermore, a facet of the optimization process worth highlighting is that not normalizing the data occasionally causes extreme outliers in the MAPE, i.e. it mostly produces much better MAPE values but occasionally an extremely large MAPE that enlarges the average, which was reflected in the stark difference between mean and median MAPE values there.

4.2 Classification

4.2.1 Bayesian optimization with 20 iterations

Evidence for these pipelines being better includes the fact that decision trees tend to be very liable to overfit their training data, so it is no surprise that pipeline changes that remove outliers and features would help it perform better. Forests are an ensemble of trees, so their behavior is somewhat related. However more features were used in the forest, because not every tree in the ensemble needs to use every feature each time. Normalization might have helped the tree for stochastic reasons i.e. it was probably equally good as not normalizing so the best pipeline having it as true is likely somewhat arbitrary.

Next, the accuracy of the tree and forest are scattered together to see if they tend to both improve together or not.

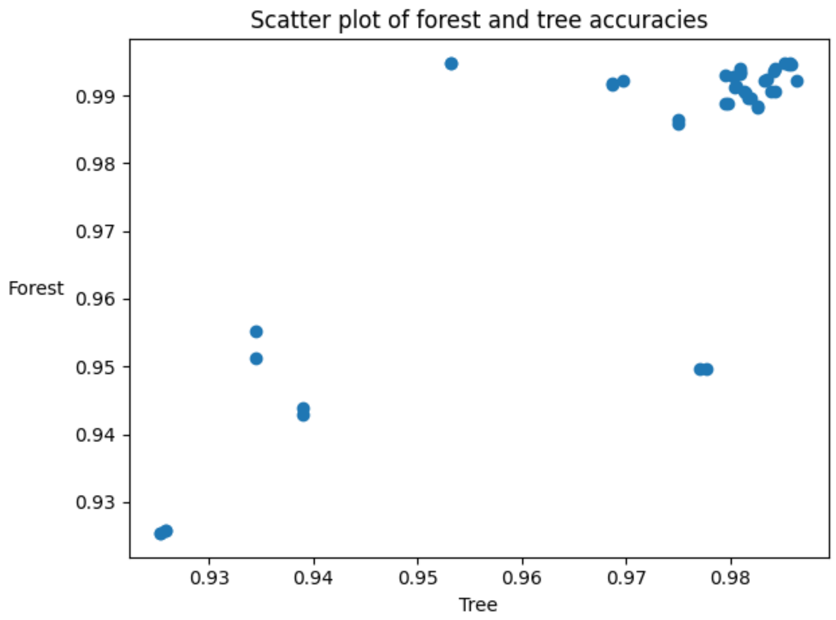

Figure 2: Accuracy scatterplot

As is seen from figure 2, a change in the pipeline was generally either good for both models or bad for both models.

The trends of optimization are shown in figure 3.

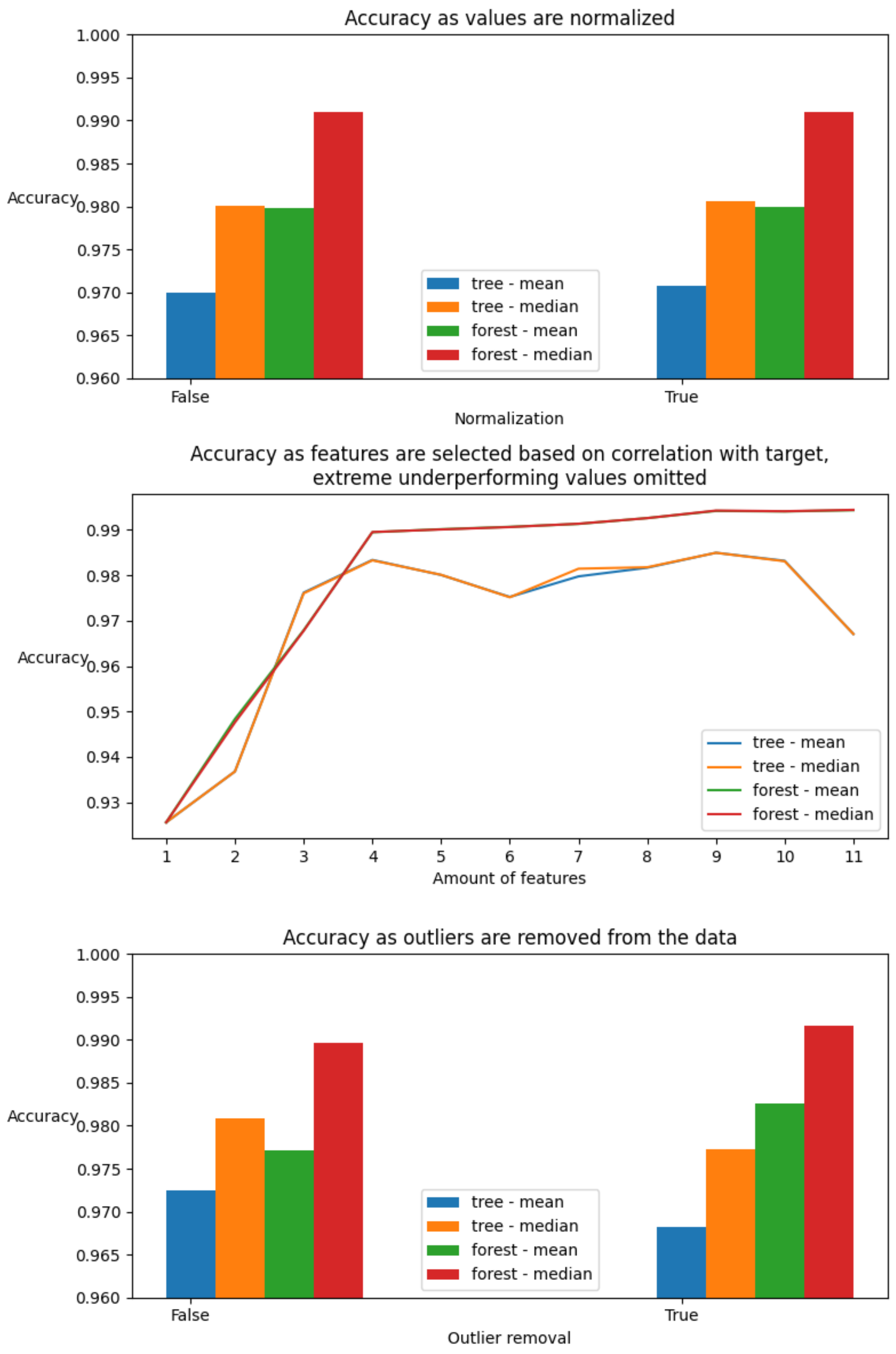

Figure 3: Classification performance trends

From the first plot, it is clear that normalization had very little affect on the tree and forest, which was a world of difference compared to what happened with linear regression. In the second plot, it is is clear that the forest benefitted from having all the features whereas the tree was slightly better off with removing a few of the features that were least correlated with the target. Third, the value of outlier removal was very model dependent, with the forest getting better but the tree getting worse.

4.2.2 Bayesian optimization with 100 iterations

Increasing the Bayesian optimization from 20 iterations to 100 iterations produced the following results (after 778 minutes!).

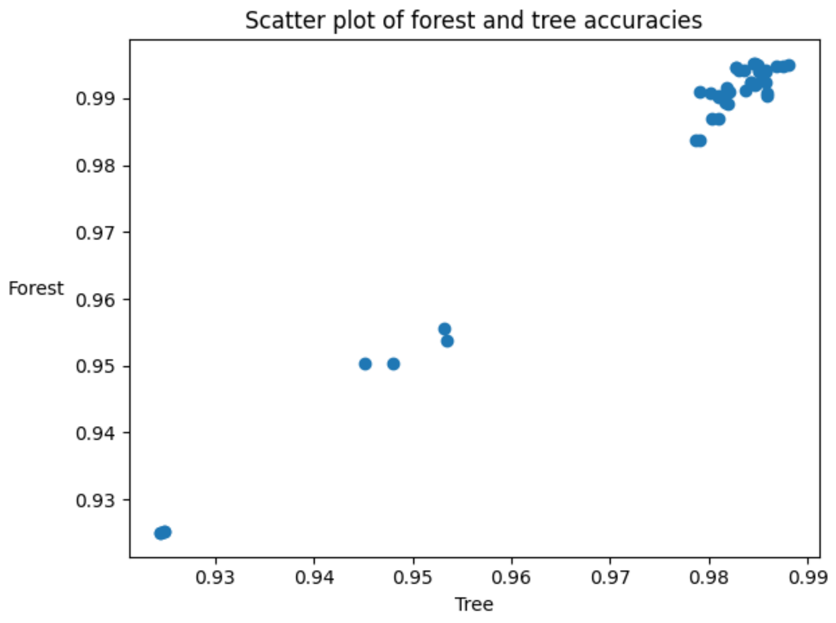

Figure 4: Accuracy scatterplot

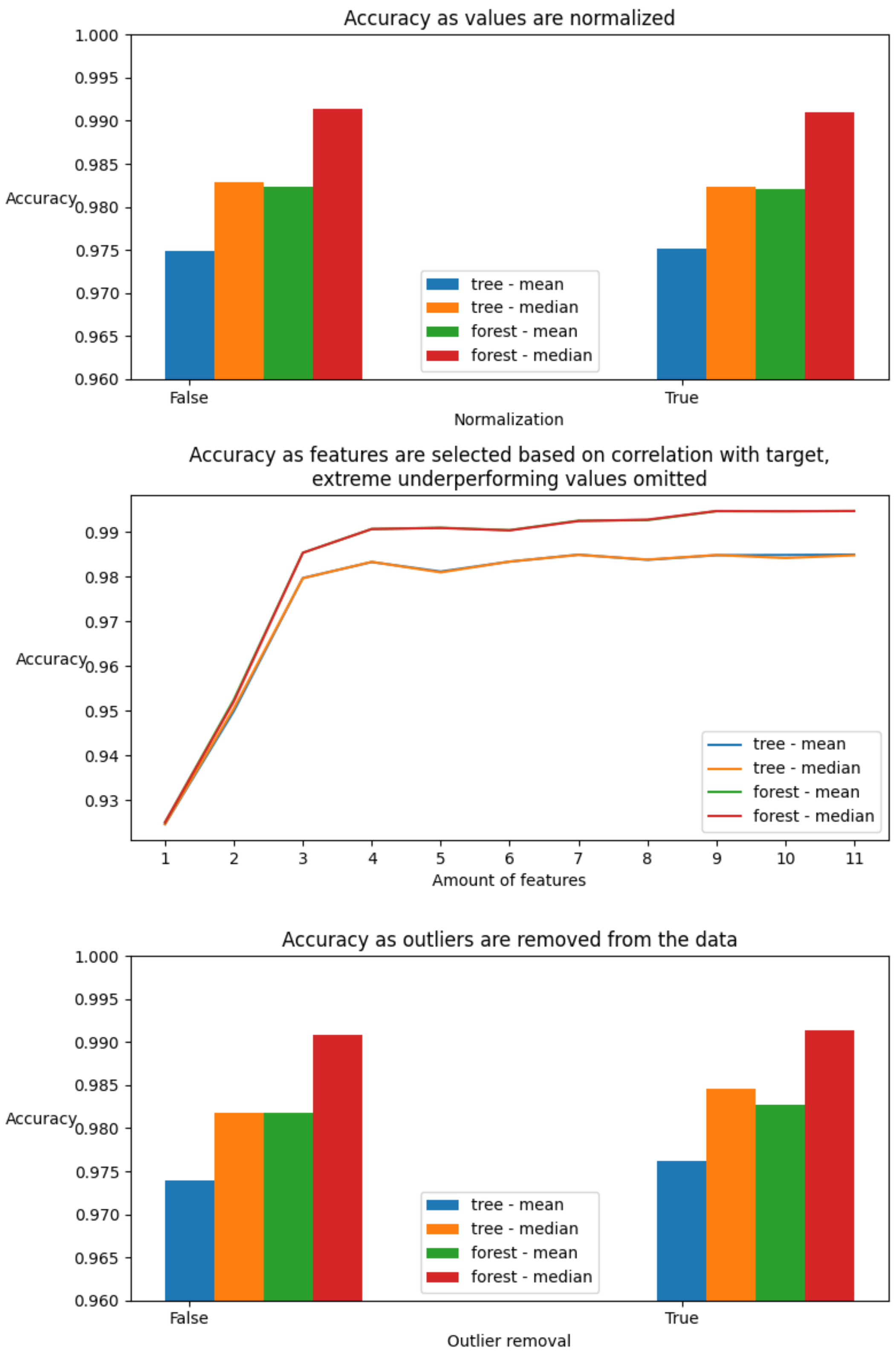

Figure 5: Classification performance trends

The results are mostly the same. What is interesting is that both models used more features now, but still not all features. Furthermore, both the tree and forest converged to the same hyperparameter settings. Tree accuracy increased a little but but forest accuracy did not increase enough to make a difference when rounded.

The main takeaways from these results is that firstly pipeline tuning has the *potential* to improve performance, but likewise is it very easy to cause harm with it. Second, when all parts are optimized together, there is a tendency for changes in the pipeline optimization to affect model hyperparameters as well. Third, the value of pipeline optimization steps are, in principle, heavily dependent on whether the situation is a regression or classification as well as the model class itself.

5 Code

The associated code is in pipe.ipynb