Outlier Removal

- I used scikit-learn's Isolation Forest, and adjusted the contamination parameter until 5 outliers were removed. (The final value for contamination was 0.013.)
- This resulted in a .11 increase in R^2 for the Bounded Lasso + LogReg model, from .53 to .64
- The data with outliers removed was also used for the following models

FFNN

Hyperparameter search was performed on a feedforward neural network regressor. I used the `hyperopt` package for Bayesian hyperparameter optimization.

Hyperparameter search space:

- **hidden layers:** uniform linear distribution between 1 and 2
- **nodes per layer:** uniform log between 3 and 100
- **learning rate*:** uniform log between .0001 and .1
- **alpha**** : uniform log between .0001 and .1
- **training epochs*****:normal distribution, mean = 150, var = 25

  *for Adam optimizer

  **l2 regularization param on input and hidden layers

  ***The values for number of epochs were a ballpark estimate from some initial experimentation/hand-tuning- 150 epochs appeared to be the number after which further improvements in test performance diminished significantly.

Unchanged parameters:

- ReLU activation on input and hidden layers
- linear activation on output layer
- All parameters for the optimizer besides learning rate were left as Keras defaults:
  - `beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False`
- Anything not described above was left as Keras defaults

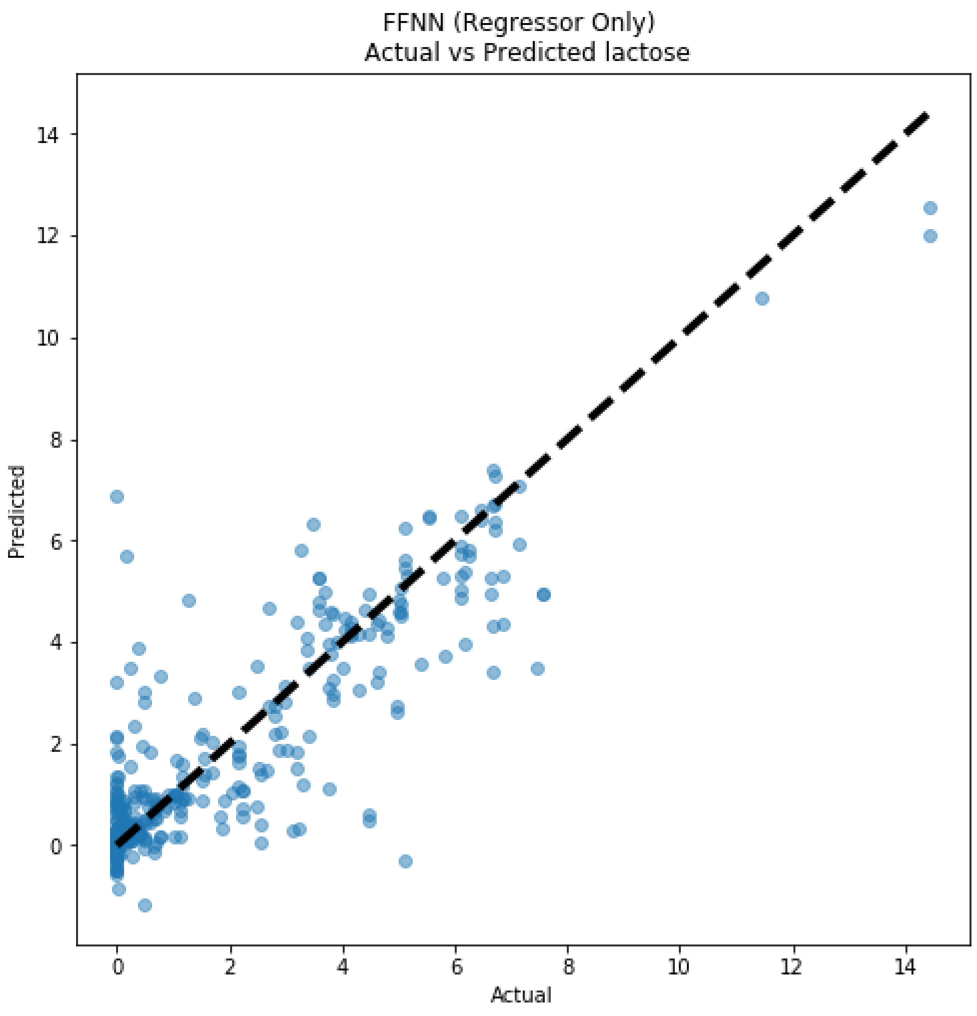
Results:

After 93 trials (I let it run overnight) the best R^2 value achieved was .77, other statistics are shown in the table at the bottom.

The parameters used for this trial were as follows:

- **hidden layers:** 2
- **nodes per layer:** 50
- **learning rate:** ~.0012
- **alpha:** ~.029
- **training epochs:** 185

Actual vs. predicted scatter plot for FFNN:



Discussion:

- **Regarding hyperparameter optimization:** Someone more experienced with training neural nets might have been able to get similar results just by hand-tuning. I chose to use Bayesian hyperparameter optimization for the following reasons:
  - Curiosity (never used it before)
  - Hand-tuning would likely have been ineffective (neural networks are sensitive to several hyperparameters, and I am not experienced in tuning them) and inefficient (lots of waiting, adjusting parameters, and waiting again).
  - Compared to other alternatives to hand-tuning (random search and grid search), Bayesian hyperparameter optimization is supposed to obtain better performance in the same number of trials by focusing on promising areas of the search space.
- **Potential Improvements:** Adding a zero/non-zero classifier and clipping negative values would probably improve performance. As shown above, the model is predicting nonzero lactose in many cases where the actual value is 0.

Gradient Boosting Model:

I also trained an gradient boosting (XGB) model using the `xgboost` package.

Regressor Only:

- Successive grid searches were used on the number of estimators and maximum tree depth.
- `col_sample_bytree` (fraction of columns to be randomly sampled for each tree) was hand-tuned.
- All other parameters were left as xgboost defaults

Results:

An R^2 value of .80 was achieved with the following parameters:

- `n_estimators` = 100
- `col_sample_bytree` = 0.9
- `max_depth` = 9

Combined Regressor and Classifier:

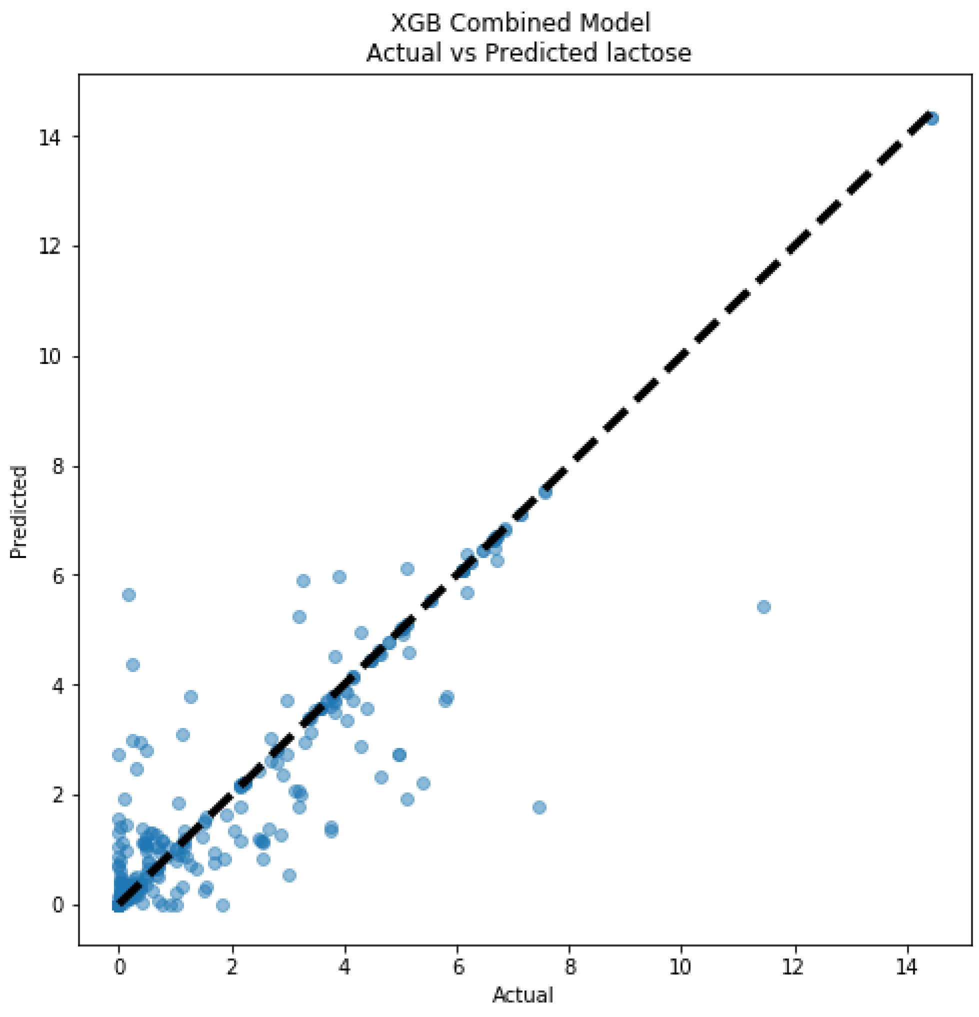
- A gradient boosting classifier was added in similar fashion to the logistic regression classifier used for the Lasso model.
- The classifier `max_depth` was tuned by hand to a value of 7 by inspecting the accuracy of the classifier on its own.
- The parameters for the regressor were left unchanged from above.

Results:

An R^2 value of .82 was achieved with the following parameters:

- `n_estimators` = 100
- `col_sample_bytree` (regressor) = 0.9
- col_sample_bytree` (classifier) = 1.0
- `max_depth_regressor` = 9
- `max_depth_classifier` = 7

Actual vs. predicted scatter plot for combined gradient boosting model:

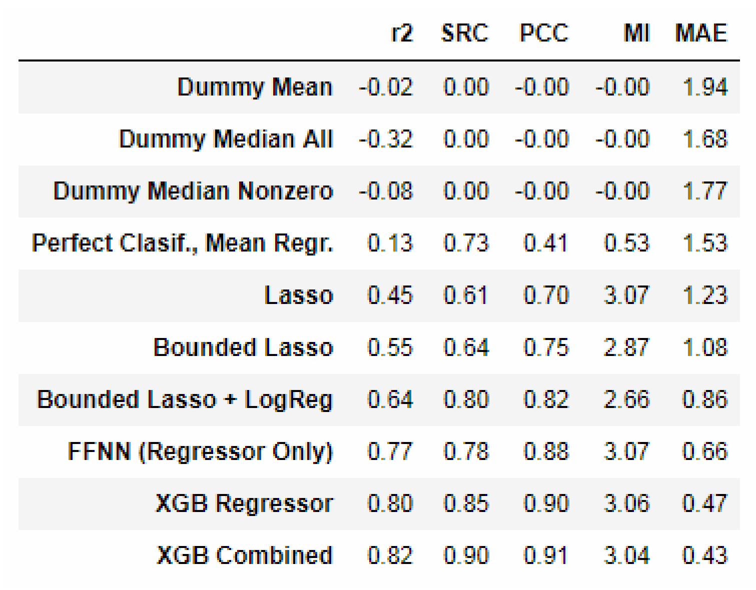


Discussion:

- These results can be considered preliminary for this model, since they are based mostly on hand-tuning. A more rigorous hyperparameter search could be conducted in the future, which could possibly lead to some slight performance increase.

Overall Results:

Statistics for all the models trained so far are shown below. The XGB Combined model currently outperforms all other models. However, the results for the FFNN and XGB models could both potentially be improved (by adding a classifier to the FFNN, and by conducting a more thorough hyperparameter search for the XGB Combined model), and it is not certain that the XGB model would continue to outperform the FFNN after these improvments.



	r2	SRC	PCC	MI	MAE
Dummy Mean	-0.02	0.00	-0.00	-0.00	1.94
Dummy Median All	-0.32	0.00	-0.00	-0.00	1.68
Dummy Median Nonzero	-0.08	0.00	-0.00	-0.00	1.77
Perfect Clasif., Mean Regr.	0.13	0.73	0.41	0.53	1.53
Lasso	0.45	0.61	0.70	3.07	1.23
Bounded Lasso	0.55	0.64	0.75	2.87	1.08
Bounded Lasso + LogReg	0.64	0.80	0.82	2.66	0.86
FFNN (Regressor Only)	0.77	0.78	0.88	3.07	0.66
XGB Regressor	0.80	0.85	0.90	3.06	0.47
XGB Combined	0.82	0.90	0.91	3.04	0.43