-Problem Description

Which of the seven given regression algorithms works best? Furthermore, what changes to the algorithms result in greater performances?

-Descriptions

K-Nearest Neighbors: Algorithm which makes predictions for the testing set based on the k nearest data points.

Linear Regression: A linear approach for modelling the relationship between an item and its attributes using a linear function of the input features to minimize mean squared error. Also goes by Ordinary Least Squares.

Ridge Regression: Similar to linear regression, in that it uses a linear function to model data, however the coefficients have a smaller magnitude to minimize the effect of the attributes.

Regression Trees: A hierarchy of if/else statements to go from observations about an item based on its attributes to conclusions about the item’s target value.

Random Forests: A extension of regression trees, which constructs multiple trees and returns the mean prediction of the individual trees.

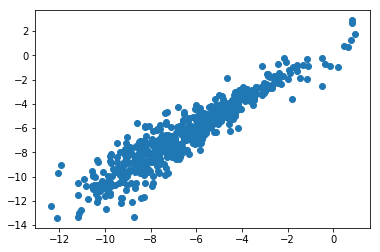
Support Vector Regression: Creates bounders along the categories of the training examples to separate data into the corresponding category, with an emphasis on creating large gaps. Employs the kernel trick to add a nonlinear feature to the representation to make the prediction more powerful.

Multilayer Perceptron: An algorithm which employs nodes that represent the input features, each using a nonlinear activation function, which together represent the weighted sum of the inputs.

-Experimental Results

K-Nearest Neighbors:

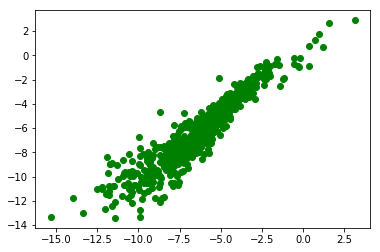
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Changes | No | Number of neighbors | Weight | Distance Used | Standardized | All applied |
| MSE | 3.560 | 3.495 | 3.446 | 3.300 | 1.074 | 1.017 |
| Gain | 0.0 | 0.065 | 0.114 | 0.26 | 2.486 | 2.543 |



Linear Regression:

No changes

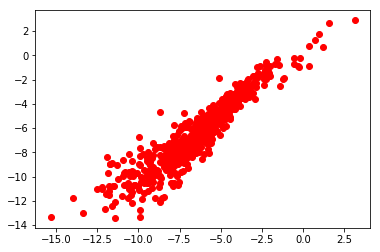
MSE 1.075



Ridge Regression:

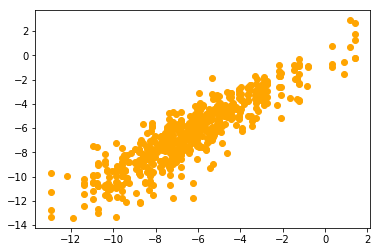
No changes

MSE 1.075



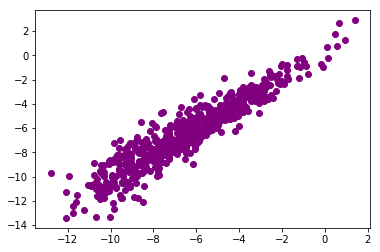
Regression Tree:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Changes | None | Depth | Min Samples split | Min samples leaf | Standardized | All applied |
| MSE | 1.917 | 1.811 | 1.743 | 1.831 | 1.702 | 1.518 |
| Gain | 0.0 | 0.106 | 0.174 | 0.086 | 0.215 | 0.399 |



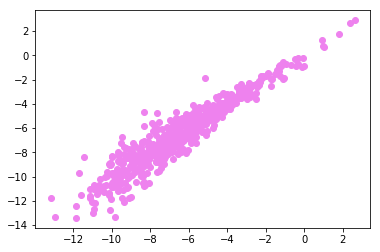
Random Forests:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Changes | None | # of estimators | Samples leaf | Max leaf nodes | All applied |
| MSE | 1.064 | 0.970 | 1.062 | 1.092 | 0.965 |
| Gain | 0.0 | 0.094 | 0.002 | -0.028 | 0.099 |



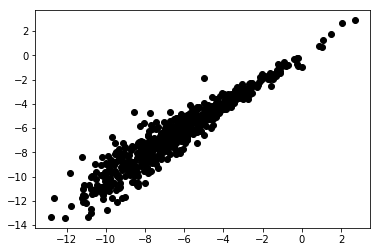
Support Vector Regression:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Changes | None | C | Epsilon | Gamma | Data Standardized | All applied |
| MSE | 7.897 | 7.866 | 7.897 | 7.248 | 0.907 | 0.801 |
| Gain | 0.0 | 0.031 | 0.0 | .649 | 6.99 | 7.096 |



Multilayer Perceptron:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Changes | None | Hidden layer size | Alpha | Beta 1 | Beta 2 | Data Standardized | All applied |
| MSE | 1.555 | 2.021 | 2.187 | 1.929 | 1.472 | 0.820 | 0.789 |
| Gain | 0.0 | -0.466 | -0.632 | -0.374 | 0.083 | .735 | 0.766 |



Analysis

K-Nearest Neighbors:

KNN performs very well for being a relatively simple and easy to understand algorithm. Standardization played the biggest role, but the other parameters did play a role in making the algorithm perform better, such as giving the distances weight and using the Manhattan distance formula.

Linear Regression and Ridge Regression:

Both linear model regression models work similarly, which in turn gave similar results. Due to the type of data being used the algorithms work very well as intended without any manipulation. Tuning the parameters resulted in worse performance most likely due to the lack of sparsity in the attributes, as well as in the small number of attributes. This in particular affects ridge regression which works to minimize these problems in linear regression, but due to there not being those problems in this data set, the results returned for both were identical up to three decimal points.

Regression Trees:

The regression tree worked well, albeit returning the worst mean squared error out of the seven algorithms. Fine tuning the attributes made the tree more responsive to the data being fed into it, and in turn minimized overfitting, allowing the predications made on the test set to be less strict returning better predictions. Despite what was went over in class the regression tree did in fact benefit from standardization while the random forests didn’t. The change isn’t major but still unaccounted for.

Random Forests:

Despite being a collection of regression trees the random forest algorithm did not in fact benefit from the same changes that the regression tree did. This is most likely due to the size of the trees being made varying and the data that goes into each tree not being the same. This randomness however is what allowed the random forest to return better predictions, as over fitting no longer becomes an issue as each tree has different data, which results in slight variations of around the same value, being averaged out into a more accurate prediction. Therefore, the most important feature in the random forest is the number of trees being used. The tradeoff then is processing time, as using one thousand trees takes substantially longer than ten and is substantially less consuming both of time and processing power then ten thousand trees. The algorithm worked very well on the data with one hundred trees as well as ten. Finally, despite the number of max leaf nodes returning a larger mean squared error when used alone, it did in fact decrease the error when used in ensemble with the rest of the changes. This is most likely due to the trees being made being more specific to the data set with all the parameters than just with the one.

Support Vector Regression:

Support vector regression becomes better as the kernel and regularization parameters become stricter, fitting the data better and creating stronger boundaries. Without standardization however, the outliers of the data will have a larger influence on the outcome, which is why the SVM benefited the most from standardization of all the algorithms. This allowed the kernel and regularization, even at the default setting to work plenty fine. Once adjusting the attributes outliers, the algorithm returns the second smallest mean squared error.

Multilayer Perceptron:

The multilayer perceptron preforms averagely at first but when tuned correctly returns the smallest mean squared error. This is despite all the changes to the parameters returning larger mean squared errors than the standard settings, except data standardization, when used alone. Standardization alone makes it second only to the support vector regression, and first once all the other parameters are applied as well. The range of the attributes and their outliers doesn’t have the biggest affect on the MLP, which is most likely why when left alone to standard use the MLP returns a MSE like that of a tuned regression tree. Once the data is standardized the activation functions work better at predicting the test data and returns the most precise of the predictions.

Conclusions:

When applicable preprocessing makes a big difference. All the algorithms that saw use from standardization made substantial improvement to their predictions, especially when compared to the other parameters that made them up. Which isn’t to say all the other parameters didn’t help fine tune all the algorithms, simply that preprocessing helps to narrow down the data and make the outliers less likely to skew the predictions, especially for the support vector. The importance of preprocessing also helps to highlight the importance of the usefulness and pitfalls on the reliance of attributes. Attributes make up the data trying to be predicted and as such are the best way to measure and predict that data but picking the right attributes to measure and ensuring that no one attribute takes all the importance in the algorithm process is key to making an algorithm work efficiently. That is why all the parameters service the processing of attributes. While seemingly obvious in nature, understanding why that is allows for better understanding of why the algorithms work the way they do, or don’t for that matter, and making sure that the data collected can actually be useful.