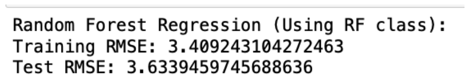
Homework 4

kvv2005

April 2024

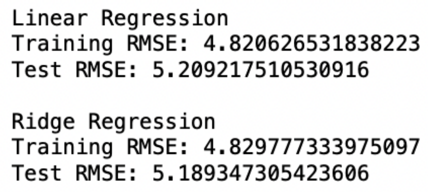
1 Random Forest

1.1



```
Random Forest Regression (Using RF class):
Training RMSE: 3.409243104272463
Test RMSE: 3.6339459745688636
```

The RF class result gives us a higher error than what I predicted. A possible reason for this is maybe I overcomplicated the class. The difference in result between the training and test data is quite small. This is good since our model seems to be performing less overfitting. The RMSE changes every time I change the number of trees, where a lower number will increase RMSE. Here, I used 80 trees.



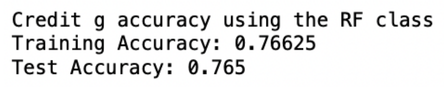
```
Linear Regression
Training RMSE: 4.820626531838223
Test RMSE: 5.209217510530916

Ridge Regression
Training RMSE: 4.829777333975097
Test RMSE: 5.189347305423606
```

Analysis:
In terms of extracting the Boston dataset and calculating ridge and linear regression, I recycled my codes from homework 2.
Linear Regression: Training and test results are relatively close to each other, which could mean that the model is adapting well to unseen data. However, the error is quite high, which could suggest underfitting. Linear regression may not be the best way to capture the data's behaviour.
Ridge: Similar analysis to linear regression, we see that the model adapts well but we still have a high error for some reason. Regularization did not aid in decreasing error.
Random Forest: Here, the difference between the training and test is high, yet still a better difference than linear and ridge. I believe

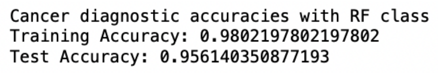
that this error can be reduced by further hypertuning of parameters. However, we can see that in general RMSE is much lower than the previous methods, which would suggest that RF is a better suited in capturing the Boston dataset. Overall, even if there are nuances within the RF implementations, they are errors than linear or ridge, so perhaps this is an appropriate classification method for this dataset.

1.2



```
Credit g accuracy using the RF class
Training Accuracy: 0.76625
Test Accuracy: 0.765
```

Using the RF class I constructed, we see that accuracy is quite low in training. However, it is quite similar to the test accuracy, which would mean that the model is consistent throughout its computations, which I believe is more important and easier to work with. It could be debated that such a result would be better than achieving 1.0 in training since the model would memorize the data and fail to classify unknown data.



```
Cancer diagnostic accuracies with RF class
Training Accuracy: 0.9802197802197802
Test Accuracy: 0.956140350877193
```

Using my RF class has probably been most successful in measuring the cancer diagnostic accuracies. Not only are the accuracies close to 1, but there is little difference between training and test. While the accuracy is not 100 for test, it is still considered quite a good result and we can say that the model effectively captures the data's behaviors.

2 GBDT

2.1

Let d= depth
m= features
n= samples
To find the best decision, it will take O(m*n) since all samples have to be analyzed with the features, which also increases as depth increases. There are around 2^d nodes in a GBDT tree. This makes our complexity to be $O(m * n * 2^d)$ which is exponential as depth increases

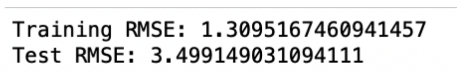
2.2

In GBDT training, the computation that would take the longest would be finding the best threshold. This is because you not only need to find the best value for this threshold, but also compute all the values corresponding to it. A non parallel solution to this could be relating to feature selection. Since initially we use all features to assess whether splitting at that node is appropriate, we can choose a subset of these features which would reduce the number of thresholds that need to be calculated.

2.3

We can use parallel computing in the finding best decision rule function in the Tree class. By using Pool.map, we separate the task of finding the best decision rule to several processes. Each process assess the best decision rule from the data it received, and all the data is pooled at the end to elect the best decision.

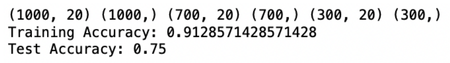
2.4



```
Training RMSE: 1.3095167460941457
Test RMSE: 3.499149031094111
```

We can see here that there is somewhat a similar performance to the random forest. It is actually performing slightly better than RF, with training having a much smaller error. There is however quite a large difference between the training and test rmse which could suggest overfitting. The model should be fine tuned to reduce this but, all in all, there is a better performance than RF, linear and ridge.

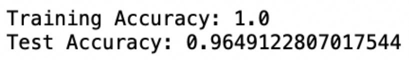
2.5



```
(1000, 20) (1000,) (700, 20) (700,) (300, 20) (300,)
Training Accuracy: 0.9128571428571428
Test Accuracy: 0.75
```

Credit-g dataset

The high training accuracy suggests that the model has properly learnt the patterns of the data. With that said, the difference between the test and training accuracies is relatively high, so the model may be overfitting the data. The issue could stem from my methodology in converting the alphabetical dataset to numerical, which probably swallowed some noise or inconsitently converted it. Also, we can also determine whether GBDT is the best possible model for this type of dataset.



```
Training Accuracy: 1.0
Test Accuracy: 0.9649122807017544
```

Cancer dataset

Here, the model performs perfectly on the training data, and nearly as well on the testing which is a very good result. It is still important to mention that reaching 100 on the training can still mean overfitting. The test accuracy shows that the model performs well to the type of unseen data that the cancer diagnostic set provides.

2.6

Overall, GBDT performs slightly better than Random forests. GBDT builds on its previous trees through boosting and its previous errors are correct by new trees. This si why we might have a higher accuracy in GBDT. Random forests are disadvantaged since they do not have access to the previous errors, and so this is why it might perform worse than GBDT in that sense. GBDT fight overfitting better than Random Forests since they produce shallow trees and later combine them. This could allow for further analysis per data which avoids overfitting. The high performance could also be due to the type of data that each set provides. Moreover, the high difference between the training and test rmse could present potential issues in analyzing more datasets, and could also show that GBDT overfits the data by memorizing the training data's patterns.