Meng Gao 24030489

Method selection

Introduction

This assignment demonstrates the issue of candidate model selection and the process of finding the best performing regression model with the Caret package with the supplied Shiny App.

Steps of model selection

1. Only focus on the methods that perform regression as this is a regression problem.
2. Taking the given period of time into consideration, try at least one from the models that share the same types or relevant characteristics but discard those that take extremely long to train as they don't tend to perform well and greatly slow down the Shiny App.
3.While choosing the regression methods, choose only one with tuning parameters as they optimize the methods' hyperparameters and perform better.
4. Choose suitable pre-processing methods, train the models and compare the performances using 'RMSE'.
5. Explore those with better performances and their similar methods based on the 'map' of regression methods.

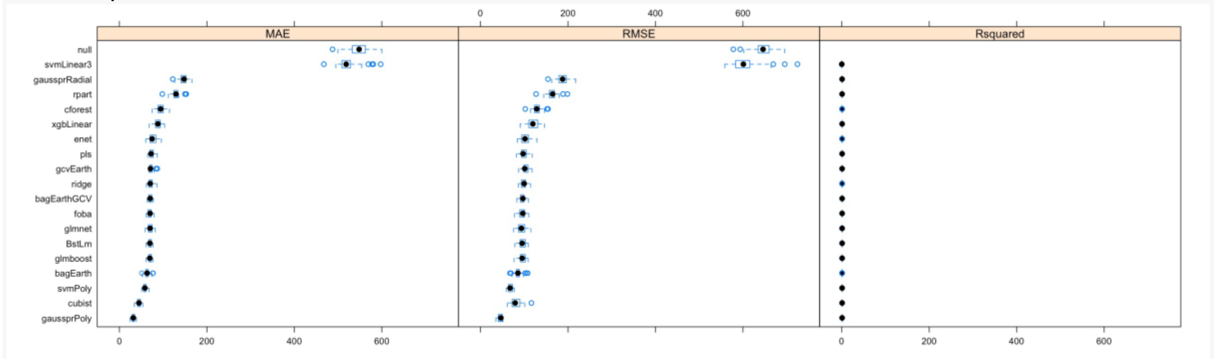
Comparison of the methods

A null model is always something we can consider first to help with the model selection, as we do not want anything worse than it. The comparison of the methods used is below. 'knnimpute' as it takes care of missing data using nearest-neighbor method, and 'dummy' helps covert factors into numeric variables. we' re using them a lot in the models. The tags are arguable as some methods have more than one feature.

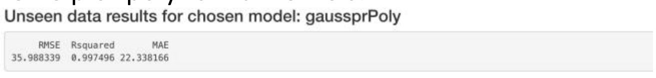
Methods	pre-processing employed	RMSE（2dp）	Tags
Null	None	643.32	None
BagEarth	knnimpute, dummy	85.81	Bagging
bagEarthGCV	knnimpute, dummy	95.80	Bagging using gcv pruning
gcvEarth	knnimpute, dummy	102.70	Multivariate adaptive regression splin
Glmboost	knnimpute, dummy,center,scale	96.13	Boosted linear
Cubist	knnimpute, dummy	81.72	Boosting (CART)
bstlm	knnimpute, dummy,center, scale	96.18	Boosted linear (CART)
xgbLinear	knnimpute, dummy	120.19	Boosting
Cforst	knnimpute, dummy	94.51	Random forest
Glmnet	knnimpute, dummy	94.24	Generalized linear model
Pls	knnimpute, dummy	99.39	Partial least squares
foba	knnimpute, dummy	95.33	Ridge regression
Ridge	knnimpute, dummy	98.85	Ridge regression
gaussprPoly	knnimpute, dummy	44.49	Gaussian
Svmploy	knnimpute, dummy	68.37	Support vector machine
Svmlinear3	knnimpute, dummy	610.03	Support vector machine
enet	knnimpute, dummy	101.82	Elastic net
Rpart	knnimpute, dummy	163.66	Tree-based (CART)

Gamboost, randomGLM, krlsPoly, gaussprRadical and a few other methods have been trained and discarded as they failed to train or the packages are not available. We also discarded logicBag, logreg as they took ages to run. Simpls, kknn and widekernelpls were abandoned as well for taking too much memory.

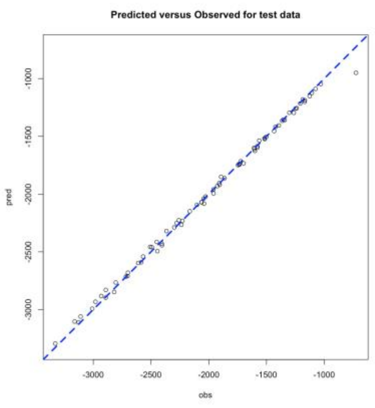
We are also interested in the model selection visualisation, which provides an accessible way to see and compare the models.



As can be seen in the graph, of all these models, the majority of them perform similarly while Svmlinear3 is almost as bad as a null model. This is a L2 Regularized Support Vector Machine (dual) with Linear Kernel. The reason why it doesn't perform well could be the data is not linearly separable.
cubist, gaussprPoly and svmploy perform the best. Bagearth's RMSE is ok but it does not look stable. The testing error of svmPoly is 54.93. This method's long name is support vector machines with polynomial kernel and it uses the polynomial kernel in the SVM models, so it allows learning of non-learning models.
gaussprPoly gives a testing error of 35.99, which makes it the best method on testing data. Below is the predicted versus actual visualisation of test data. This method is a gaussian process with polynomial kernel which performs well on both training and testing data in this dataset. The idea of a gaussprPoly model is basically using the models that explicitly assume the function has a particular parametric form in addition to a Gaussian Process model(Schulz, Eric, Maarten Speekenbrink, and Andreas Krause). The reason why it performs well is probably because it takes care of non-linearity with the help of polynomial kernels.

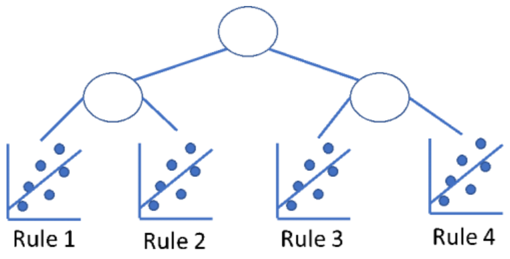

Unseen data results for chosen model: gaussprPoly

```
     RMSE  Rsquared       MAE
35.988339  0.997496 22.338166
```



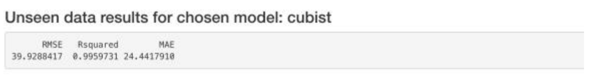
Cubist method has a testing error of 39.93, while its RMSE for training data is the largest of the three models. This could be due to overfitting or its instability. As can be seen in the boxplot of RMSE, it has bigger whiskers. This method is a combination of tree-based method and regression method. With a framework of a tree, each terminal leaf contains a multivariate regression model. The picture below demonstrates how it works.

Meng Gao 24030489



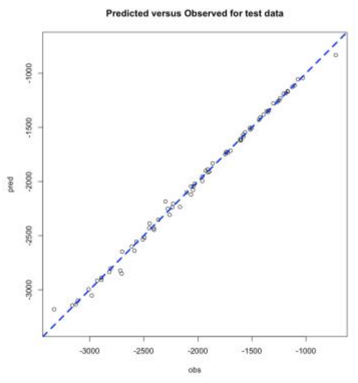
(Science, ODSC - Open Data)

The rulequest.com describes cubist as a powerful tool for generating rule-based models that balance the need for accurate prediction against the requirements of intelligibility. Cubist models generally give better results than those produced by simple techniques such as multivariate linear regression, while also being easier to understand than neural networks(rulequest.com). If we're looking for a transparent model, cubist could be the best choice. The predicted versus actual visualisation of test data is as follows.



Unseen data results for chosen model: cubist

```
     RMSE   Rsquared        MAE
39.9288417  0.9959731 24.4417910
```



Discussion

The methods perform better with the help of fining tuning, one thing that these algorithms have in common is they can detect interactions and linearities. If transparency is important, we'll need to choose a method with fine performance and interpretability. If we take scalability into consideration, cubist could be the best method as it's much faster than SVM methods generally.

One of the scenarios that we need ensembled methods could be one model has good performance but does not deal well with numerous observations, we might consider ensemble it with ones that can handle big datasets. But we don't ensemble the methods with predictions that are highly correlated with each other. When the predictions are less correlated with each other, the ensembles tend to perform better (Prabhakaran, Selva).

Conclusion

By applying and comparing over 20 different models, this assignment demonstrates the idea of "no-free lunch" - there is no method that works optimally in all situations. With fining tuning, the algorithms that can detect interactions and linearities perform better compared with some genetic methods.

Constraints

Adding extra columns with non-linear functions and/or interactions based on the existing columns might be a good choice, as the best method we found takes care of non-linearity and interactions.

Reference

Meng Gao 24030489

1. Science, ODSC - Open Data. “Balancing Interpretability and Predictive Power with Cubist Models in R.” *Medium*, Medium, 5 Dec. 2019, medium.com/@ODSC/balancing-interpretability-and-predictive-power-with-cubist-models-in-r-858d2c936b79.
2. “Data Mining with Cubist.” *Information on Cubist*, rulequest.com/cubist-info.html.
3. Prabhakaran, Selva. “Caret Package – A Practical Guide to Machine Learning in R.” *Machine Learning Plus*, 20 May 2018, www.machinelearningplus.com/machine-learning/caret-package/.
4. Schulz, Eric, Maarten Speekenbrink, and Andreas Krause. "A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions." *Journal of Mathematical Psychology* 85 (2018): 1-16.