Finding the appropriate regression model using caret

1 About the data

There are 380 observations of 20 variables in the data set. The outcome variable is numeric, so it is a regression problem. Among the 19 predictors, only BloodType is nominal while others are numeric. The radio of observation counts to the number of predictors is 20, which is larger than 10. Hence, there is no need to worry about the model overfitting problem.

The Pearson correlation matrix of numeric variables in the correlogram shows some dimensions are highly correlated, which are shown in Table1.

Table1 Group of highly correlated variables

Group	variables
Group1	EM
Group2	KB
Group3	ACG I
Group4	DFHJ N

Data has correlated predictors; accordingly, we should potentially remove variables that have large absolute correlations with other variables before building a linear regression model. Besides, some methods like enet or glmnet which can deal with highly correlated independent variables are also optional.

Generally, “an absolute correlation coefficient of >0.7 among two or more predictors indicates the presence of multicollinearity”(RekhaMolala).Potential solutions would be removing some of the collinear independent variables or choosing regularization methods such as pca or pls which are analysis methods designed for highly correlated variables and are competent to control the model complexity as well.

The preliminary visualization demonstrates 14 predictors have missing values to some extent. As we could not determinate the types of missing data, the safe way is to utilize methods like rpart that is intrinsically tolerant of missing data. Imputation is an alternative strategy and knnimpute is utilizing in this case.

Based on a default IQR multiplier value of 1.5, the box plots show potential outliers for 14 independent variables. Correspondingly, methods like qrf, rlm which are intrinsically tolerant of outliers are underlying candidates.

2 Split the data

The function createDataPartition() is used to split test/training partitions. 80% of the data goes to training and 76 observations are left for testing.

3 Choose candidate algorithms and train the models

In general, there are four styles of methods: Neural networks, OLS related, tree-based, and kernel methods. I decided to choose some methods from each style with consideration to the problem constraints, and then concentrate further choices around those with good performance.

We started from the simple linear regression lm at the beginning. Due to the presence of outliers and correlated predictors, I tried rlm and lmStepAIC respectively. The regression assumptions are not verified very well, so I turned to several generalized linear models like glm, bayesglm, and glmnet. As some predictors are highly correlated, I also tried pls or using pls or ica during data preprocessing.

As for neural networks, neuralnet is selected as the representative. However, its performance is worse than the NULL model, so neural networks are discarded in this case.

In terms of kernel methods, SVM is the handy method and different kernels were chosen(svmLinear2, svmPoly and svmRadial) individually. Finally, svmPoly has minimal RMSE which means a polynomial model would be a better choice and we could concentrate further choices around polynomial models. Then, I selected another two models gaussprPoly and krlsPoly. krlsPoly has a terrible performance while the gaussprPoly really surprised me! Its RMSE is almost half of that for other methods I have tried so far.

Since polynomial models had better performance, why could not we try some generalized additive models or multivariate adaptive regression methods? Here we chose gam, gamboost, earth, bagEarth, and gcvEarth, but these methods just showed ordinary performance and some of them are time-consuming. BagEarth did not train successfully on any sets of resampled train data so we abandoned it and moved on.

As tree-based models are easy to represent visually, and the random forest is an example of a general-purpose method, I decided to select some methods from this aspect. Rpart, blackboost, RRF,qrf, and xgbTree were chosen but we still did not reach a superior model than gaussprPoly. We discarded xgbTree because its training time was even beyond tolerance.

Finally, I tried two prototype models: cubist and knn. Knn had an awful performance but the RMSE of cubist is great.

The details are shown in Table 2 in the appendix.

4 Select optimal models

In Caret, model tuning is automated and utilizes resampling. In this case, the bootstrap method is used for resampling, and the number of resampling iterations is 25. Assuming "best" means "minimum RMSE", the metric distribution for each model is shown in Figure1(Only 15 models are saved in the "SavedModels" folder, while others without good performance or those would slow the startup are cleaned out).

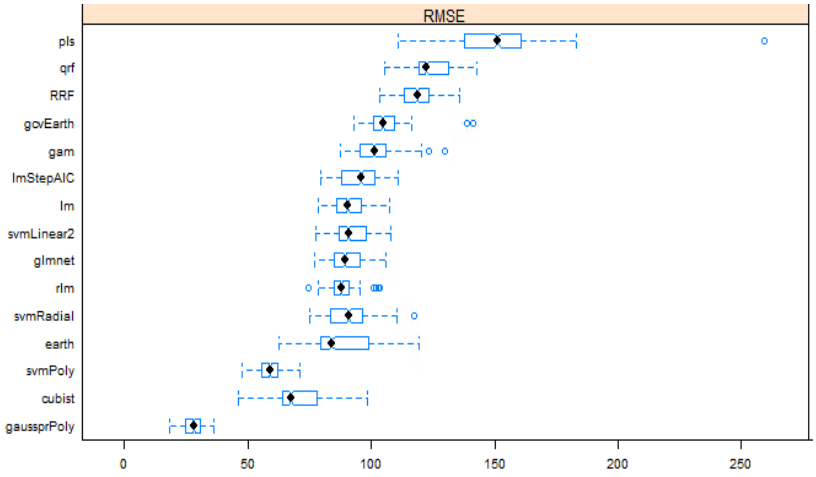

Figure1 The boxplot of RMSE distribution

According to Figure1, gaussprPoly has the minimum RMSE on average with a narrow confidence interval for the median. Besides, its spread is the smallest and the minimal RMSE is inside the zone of investigation OR that it is rapidly approaching a slope of zero(Figure2). Additionally, its computation time is not long(Table 2 in the appendix).

Resampled performance:

degree	scale	RMSE	Rsquared	MAE	RMSESD	RsquaredSD
2	0.28	27.81	1.00	17.66	4.37	0.00

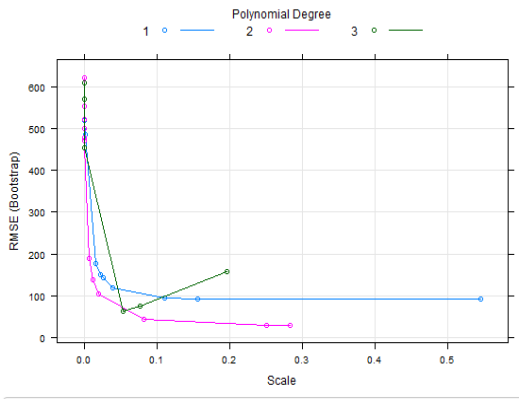

Figure2 gaussprPoly:model tuning

Unseen data results for chosen model: gaussprPoly

RMSE	Rsquared	MAE
9.4057972	0.9998051	6.7882564

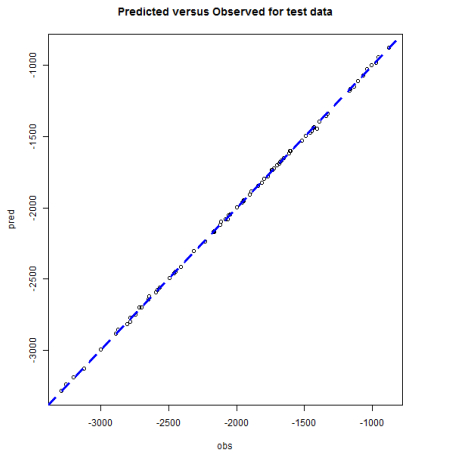

Figure3 gaussprPoly: predicted vs actual

The second choice would be svmPoly. I do not recommend cubist and earth due to the wider spread and if we check the model plot of earth, we could find its best hyper-parameters on the edge of the grid, which means that earth is not a stable method for this data.

If transparency is very important, a linear regression model would be an optimum model. The rlm has relatively small RMSE and it is fast(Table 2 in the appendix). But the outliers in the boxplot indicate rlm occasionally performs atrociously. Instead, lm would be an alternative choice among the models I have trained. The notches of lm and glmnet overlap so we are 95% confident that the two models are not statistically distinguishable. But the interpretability of lm is higher than that of glmnet.

Back to the gaussprPoly model, we need to assess the accuracy of the best model using the test data in the end. The predicted versus actual visualization of test data is shown in Figure3. The points of predicted and actual values of test data follow a 45-degree line and the RMSE is only 9.41.

On one hand, the key benefit of Gaussian processes is that “the uncertainty of a fitted GP increases away from the training data”(Knagg, Oscar). In regression, GP directly gives us a distribution for the prediction value rather than just one value. On the other hand, our data are correlated, and it is very difficult to fit a linear regression line with

a low RMSE. It is reasonable to assume that the relationship is not linear, So we can attempt to fit a gaussian process model with a polynomial kernel.

When it comes to ensemble models in machine learning, they “combine the decisions from multiple models to improve the overall performance”(Smolyakov, Vadim) and they could capture the linear and the non-linear relationships in the data, enable to minimize noise, bias, and variance, and get a better and more stable prediction. All grouped models should have similar performance and have either uncorrelated or negatively correlated residuals. If the methods with “ensemble tag” or have positive correlations of residuals, they are not suitable for assembling as that could not increase the performance of an ensemble. Furthermore, due to the high complexity, model interpretability will reduce, so it is difficult to draw any crucial business insights(Juhi). Finally, the duration for design and computation will be longer hence ensemble models are not appropriate for real-time applications.

Works Cited

Knagg, Oscar. “An Intuitive Guide to Gaussian Processes.” *Medium*, Towards Data Science, 15 Jan. 2019, towardsdatascience.com/an-intuitive-guide-to-gaussian-processes-ec2f0b45c71d.

Schulz, Eric, et al. “A Tutorial on Gaussian Process Regression: Modelling, Exploring, and Exploiting Functions.” *Journal of Mathematical Psychology*, vol. 85, 2018, pp. 1–16., doi:10.1016/j.jmp.2018.03.001.

Juhi. “Simple Guide for Ensemble Learning Methods.” *Medium*, Towards Data Science, 6 Mar. 2019, towardsdatascience.com/simple-guide-for-ensemble-learning-methods-d87cc68705a2.

Smolyakov, Vadim. “Ensemble Learning to Improve Machine Learning Results.” *Medium*, Stats and Bots, 7 Mar. 2019, blog.statsbot.co/ensemble-learning-d1dcd548e936.

RekhaMolala. “Correlation and Collinearity - How They Can Make or Break a Model.” Medium, Clairvoyant Blog, 23 Mar. 2020, blog.clairvoyantsoft.com/correlation-and-collinearity-how-they-can-make-or-break-a-model-9135fbe6936a.

Appendix

A: The table of the methods, pre-processing employed, resampling results

Table 1 the methods, pre-processing employed, resampling results (tuneLength = 30)

No.	Method	Pre-processing	RMSE	Hyperparameters
0	null		631.05	parameter=none
1	lm	knnimpute lincomb corr	91.38	intercept=TRUE
2	rlm	corr	88.68	interceptt=TRUE psi= psi.hampel
3	lmstepAIC	knnimpute	94.84	parameter=none
4	glm	knnimpute nzv corr center scale dummy	92.67	parameter=none
5	bayesglm	knnimpute corr center scale dummy	93.67	parameter=none

6	glmnet	knnimpute center scale dummy	89.97	alpha=0.50 lambda=4.05
7	pls	knnimpute dummy	153.60	ncomp=19
8	svmLinear2	knnimpute dummy	91.93	cost=0.08
9	svmPoly	knnimpute dummy	58.78	degree=2 scale=0.02 C=30.73
10	svmRadial	knnimpute dummy	91.22	sigma=0.01 C=4.29
11	**gaussprPoly**	**knnimpute dummy**	**27.81**	**degree=2 scale=0.28**
12	krlsPoly	knnimpute dummy	628.21	lambda=0.27 degree=2
13	gam	knnimpute	102.91	select=TRUE method=ML
14	gamboost	knnimpute	96.64	mstop=213 prune=no
15	earth	knnimpute	88.84	degree=2 nprune=8
16	gcvEarth	knnimpute	107.40	degree=1
17	rpart		159.23	cp=0.00
18	blackboost	knnimpute	102.55	maxdepth=3 mstop=879
19	RRF	knnimpute dummy	119.22	mtry=15 coefReg=0.80 coefImp=0.86
20	qrf	knnimpute dummy	124.88	mtry=9
21	knn	knnimpute center scale dummy	539.05	k=16
22	cubist	knnimpute dummy	69.39	committees= 69 neighbors=1
23	neuralnet	knnimpute dummy	630.73	layer1=8 layer2=0 layer3=5

B: The screenshot of the model selection visualization

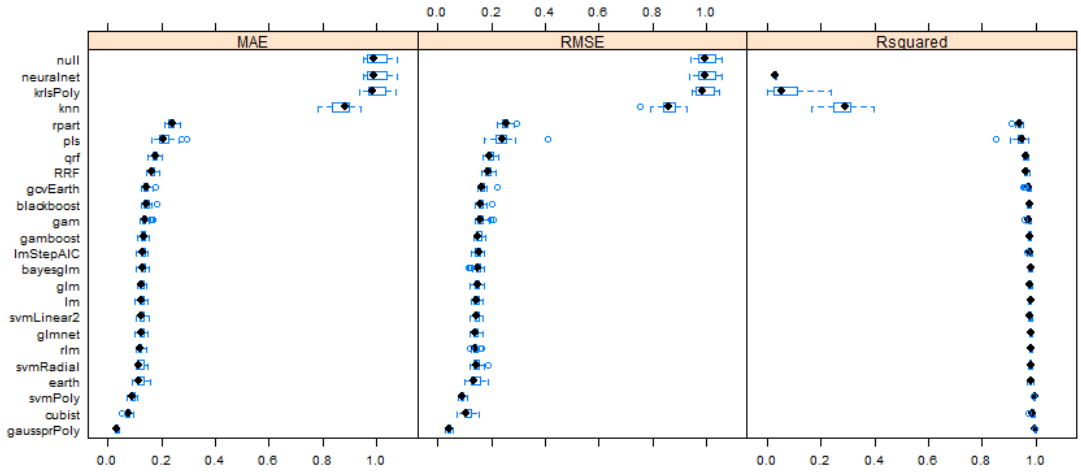

Figure2 The screenshot of the model selection visualization

C: Timings of models

Table 2 Timings of models

No.	Method	timings: Everything	timings: FinalModel
0	null	42.92	0.03
1	lm	44.27	0.18
2	rlm	28.54	0.07
3	lmStepAIC	39.13	0.29
4	glm	54.23	0.33
5	bayesglm	52.37	0.3
6	glmnet	73.56	0.27
7	pls	38.38	0.19
8	svmLinear2	184.86	0.19
9	svmPoly	73.31	0.19
10	svmRadial	69.36	0.87
11	gausssrPoly	68.53	0.23
12	krlsPoly	71.53	0.31
13	gam	1250.63	16.41
14	gamboost	73.14	0.77
15	earth	43.72	0.23
16	gcvEarth	51.1	0.2
17	rpart	25.81	0.07
18	blackboost	167.28	6.84
19	RRF	132.61	1.95
20	qrf	70.83	0.86
21	knn	72.12	0.23
22	cubist	116.1	1.64
23	neuralnet	178.66	0.95