Megha Swarini Sinha

71189549

dsi Assignment 3

**Dataset-**

On checking the dataset with help of boxplot, data table, and correlation plot we can see that though there are few outliers however these are not present when we increase the IQR multiplier to 2.5. However, in general multivariate models work well with the outliers. In this case we need not think much about outliers.

There are missing values present for the columns DoseA to DoseN.

Dose L, F, J, H, D, N are highly positively correlated to each other whereas Dose A, I, C, G are highly positively correlated to each other. Dose M and E are highly positively correlated and Dose B and K are highly positively correlated.

Variable Y has high negative correlation with variables Dose A, I, C, G. Y has mostly negative values.

**Model Selection-**

For the optimal model selection, we try to minimise the MSE, RMSE and MAE and maximise for any other metric. However, in general model metric used for regression problems are RMSE, MSE and r2

Another way to choose the candidate model is prediction of speed, complexity of the model and transparency of the model.

The **Null Model** is the base model for the observation which does not apply any algorithm and makes prediction based on averaging the values. Thus, if we want to predict y, it will train model in such a way that will give output just as an average value of y.

In the first step towards model selection I have chosen models only which is good for regression problems and perform better than null model. Null model is the baseline which predicts fixed mean value of the outcome.

The Null Model gives us value of RMSE as 747 approximately.

After running the null model, our target should be to get rmse at least more than null model.

**Models given-** The models which were already provided were GLMNet, PLS and Rpart.

Checking the performance of each of the models-

1. **GLMNet:**

RMSE of this model is nearly 180 for training set. It uses the pre-processing of naomit for omitting the na values as it does not handle the missing values and dummy to handle factor variables. The hyperparameters for this model is alpha and lambda. Best tuning parameters are alpha = 0.10 and lambda = 5.60.

Alpha is mixing percentage and lambda is a regularization parameter.

It fits a generalized linear model via penalized maximum likelihood. The algorithm is extremely fast, and can exploit sparsity in the input matrix.

This model does not do feature extraction. It does L1 and L2 regularization using lambda value. It hence works well when having large number of features as it helps in creating parsimonious model using regularization technique.

R-square method is used for calculating how good the model is from the baseline. And generally used as it is scale free. In this case we have other models to compare with hence we can check both rmse and r-square values.

For glmnet r-square is close to 1 which means, it is performing far better than the baseline model.

1. **PLS Model:**

Time taken by PLS is 18.98 sec. We have applied pre-processing such as KNN-impute to take care of the missing values and dummy to take care of factor variables. Time taken by the PLS model is almost same as GLMNet however the RMSE for GLMNet is very less compared to the PLS model, for this dataset PLS is not a good option to choose though we have large number of variables.

1. **Rpart Model:**

R-part is a formula interface hence pre-processing is not necessary for this. The transformation of log can be applied for formula interface models.

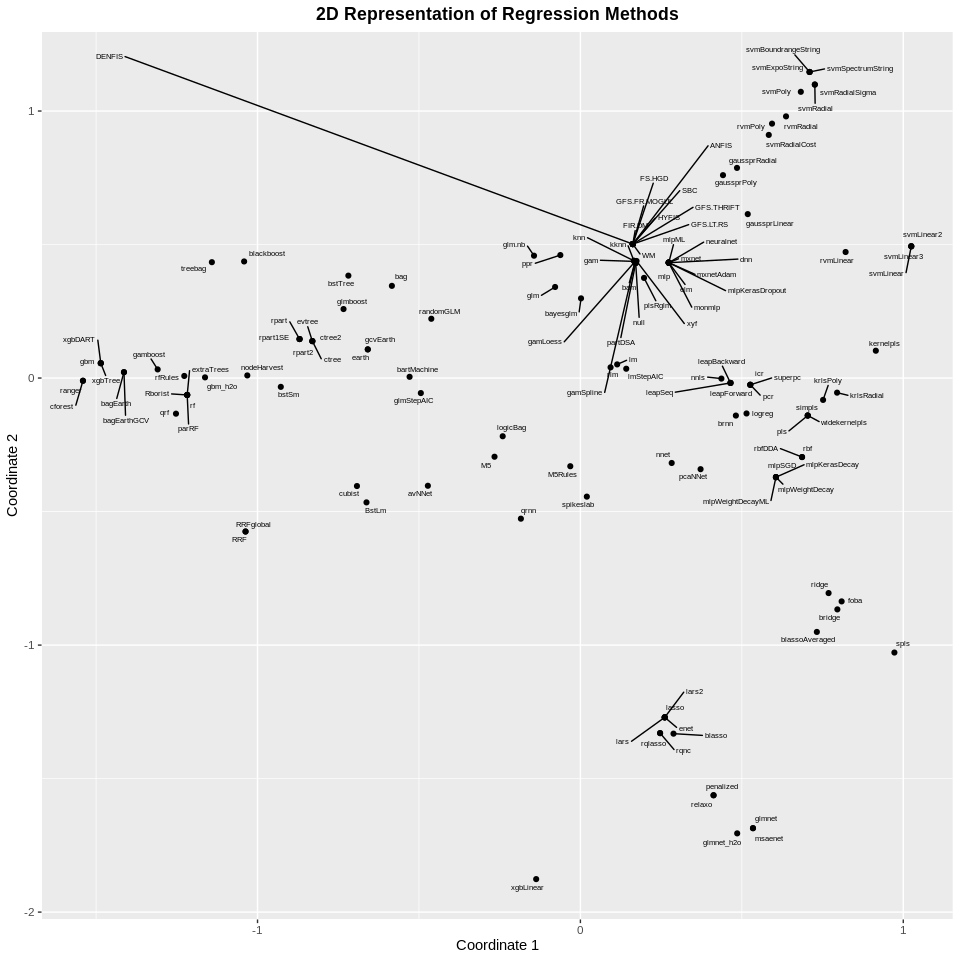
R does not store information about formula effectively so the computation in general slows down for formula interface. However, in this case the training time for rpart is less compared to PLS and GLMNet model.

It does not require any pre-processing as it can handle missing values and factors. The performance of this model is better than PLS as seen in the box-plot. It has only one hyperparameter that is CP.

**Things I considered for model selection:**

Training time available: As the training time is limited hence, we cannot apply brute force approach. Thus, narrowing down the models is required. As we have been given regression problem hence, we can choose only the regression models from list of all the caret models.

I tried several approaches for the model selection among which one was taking models from different families such as choosing tree-based model, kernel methods, generalized linear model, models using Bayesian probability or gaussian process, rule-based ensemble models and neural net. For finding the models from different groups I tried looking at the model map shown below and took models which has far distance from each other such as gbm (tree based), rvmPoly (Polynomial kernel and Bayesian model), bayesglm (Generalized Linear model), cubist (rule-based model) etc. On trying gbm, I found tree-based model was working well thus tried using random forest and it worked better than gbm. However, I also checked rvmPoly which was working well, further tried changing kernels and found result is best for polynomial kernel. Hence, I tried to work on few more models which has polynomial kernel and based on gaussian process which is gaussprPoly. This model was performing better than rest of the models if we consider rmse as metric. I also tried Rule-based model such as cubist and neural net model such as brnn which were performing.



I also checked the few models which does regularization such as ridge and lasso but did not give a better result compared to the gaussian model.

As the tree-based model such as randomforest and gbm does implicit feature selection, I thought it will outperform but it did not work well.

Though cubist is performing well in this case but in general rule-based model is used where there are some problems that are in a way inherently “data-less” where past data doesn't apply and the predictions rest primarily on intuition. In these situations, rule-based models are in general preferred. As we are not well aware of the problem statement hence, we cannot use this model.

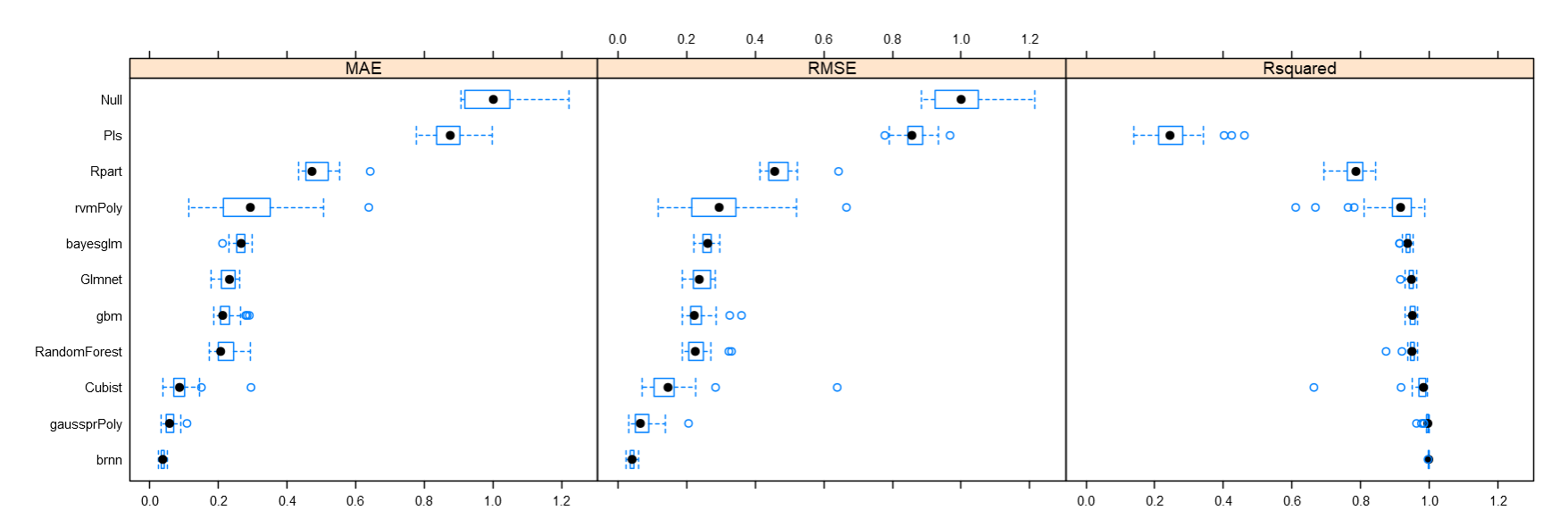
For this dataset basically, kernel method based on Bayesian statistics is working well because these models use prior probability distribution.

**Missing value strategy employed:**

|  |  |
| --- | --- |
| **Model Names** | **Missing value Strategy applied** |
| GLMnet | naomit |
| PLS Model | knnimpute |
| R part | Not required |
| bayesglm | Not required |
| gbm | Not required |
| gaussprPoly | naomit |
| rvmPoly | bagimpute |
| RandomForest | knnimpute (does not work with naomit) |
| Cubist | knnimpute |
| Brnn (neural net) | bagimpute (does not work with the naomit) |

Few methods do not need missing values to be taken care as they can handle them. Methods like Rpart, bayesglm and gbm performs even if they have missing values. Though bagimpute takes a lot of time compared to naomit or knnimpute, it is giving good result for rvmPoly. In general imputation method should be used over the naomit if the dataset is too small and there are lot of missing values. In this case, if we are omitting the rows with NA, there are 210 rows left out of 280 rows. If we consider (No. of Observation)/ (No. of variables) we are getting more than 10. Moreover, we are getting good values for rmse when using naomit with gaussprPoly model compared to knnimpute, whereas bag impute is taking lot of time (on an average 5-10 min). Decrease in rmse because of knnimpute might happen if the imputation method is producing some rubbish results. Thus, I have chosen naomit for gaussprPoly as it is giving good result with this approach. However, few methods like brnn does not work well with naomit and hence we need to do bagimpute for this as knnimpute is not performing well. To conclude from above we can say that knnimput is not able to perform imputation well on this dataset hence, we can use bagimpute but again bagimpute is taking lot of time to train, so if time is the factor and method is giving good result with naomit, then we can go with naomit method. I left knnimpute method in few of the models as there was not much change in performance for these models with the change in imputation process. Example cubist and randomforest performs equally well with naomit, bagimpute or knnimpute. Medianimute is not giving good result for any of the models.

**Screenshot of the model selection visualization:**



By default, the model selection has been fixed to gaussprPoly as this is the best model chosen by me.

**Table of the methods and pre-processing employed:**

For Non-formula interface, transformation needs to be made before using the factors or dummy variables.

|  |  |
| --- | --- |
| **Model Names** | **Pre-processing applied** |
| GLMnet | naomit, dummy |
| PLS Model | Knnimpute, dummy |
| R part | No pre- processing required |
| bayesglm | Working fine without pre-processing |
| gbm | dummy |
| gaussprPoly | naomit, dummy |
| rvmPoly | bagimpute, dummy, center, scale |
| RandomForest | knnimpute (naomit does not work) |
| Cubist | knnimpute, dummy |
| brnn | Bagimpute, dummy |

I have not chosen nzv for any of the model pre-processing as the variance of the predictors in our dataset is not close to zero.

Yeo-Johnson is a [data transformation](https://en.wikipedia.org/wiki/Data_transformation_(statistics)) technique used to stabilize variance, make the data more [normal distribution](https://en.wikipedia.org/wiki/Normal_distribution)-like. Its speciality is that it allows transformation for zero and negative value as well. However, in this case Yeo-Johnson pre-processing does not provide any improvement to the results for any of the models.

Most of the methods needs factor variables and missing values to be taken care of. However, rpart model does not require any of these.We do not need to do standardization for this dataset as the boxplot shows that there is not much difference in scales of the dataset. Thus, center and scale pre-processing is not required in most of the method except for rvmPoly.

**Tuning Parameters for each model:**

|  |  |
| --- | --- |
| **Model Names** | **Tuning parameters** |
| GLMnet | Alpha, lambda |
| PLS Model | ncomp |
| R part | cp |
| bayesglm | none |
| gbm | n.trees, interaction.depth, shrinkage, n.minobsinnode |
| gaussprPoly | Degree, scale |
| rvmPoly | Degree, scale |
| RandomForest | mtry |
| Cubist | Committees, neighbors |
| brnn | neurons |

Gbm has the greatest number of tuning parameters among the candidate models chosen. Bayesglm has no tuning parameters hence we can say it is very simple and transparent model. GaussprPoly and rvmPoly both has two hyperparameters to tune whereas neural net brnn also has one hyperparameter.

**Time taken by each model:**

|  |  |
| --- | --- |
| **Model Names** | **Time Taken** |
| GLMnet | 36.49 sec with knnimpute method |
| PLS Model | 13.71 sec |
| R part | 6.61 sec |
| bayesglm | 7 sec |
| gbm | 17.81 sec |
| gaussprPoly | 27 sec |
| rvmPoly with bagimpute | 593 sec |
| RandomForest | 33.83 sec |
| Cubist | 40.97 sec |
| Brnn with bagimpute | 214.25 sec |

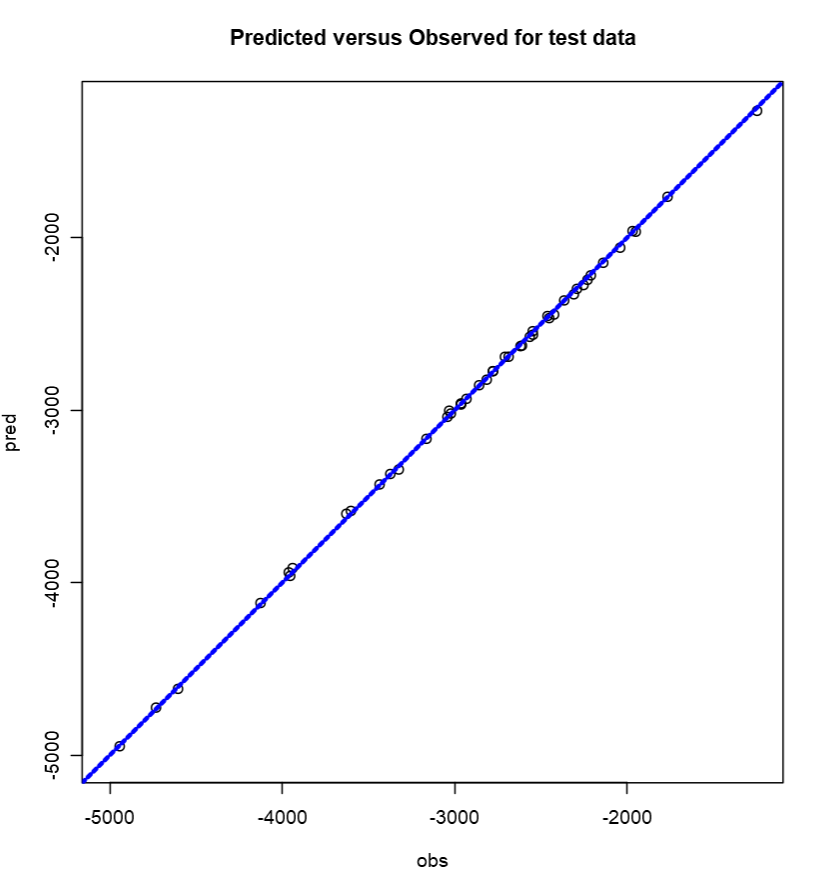
**Statement about the best performing model:**

The best performing model with respect to the rmse value is gaussprPoly and brnn(neural net) as it is giving rmse of 11 approx. For gaussprPoly there are only two parameters to tune which are degree and scale whereas for brnn we have 92 parameters to estimate. Though brnn is performing well, we can use gaussprPoly method as it is **more transparent and takes less time comparatively**. Hence, we can say there is tradeoff between good performance and computational cost. **GaussprPoly** model with naomit method for missing value, rmse is good, computational cost is less and time taken is also less thus I have chosen gaussprPoly as my best model. If we would have used bagimpute for gaussprpoly, it would have given good rmse as well but time taken will be approx. 10 min.

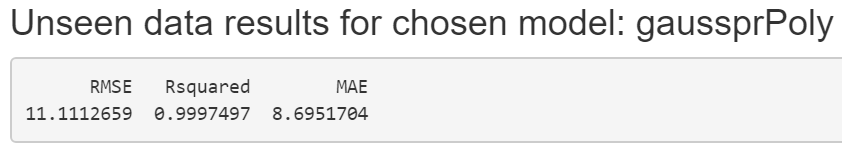
The model is using gaussian probability process to make prediction which performs well as it takes care about the uncertainties of the dataset and polynomial kernel is well suited for this dataset as the polynomial kernel not only looks at the given features of input samples to determine their similarity, but also combinations of these. These are called as interaction features.

Uncertainties in the dataset can be represented as a set of possible outcomes and their respective likelihood (probability distribution). Any data given to us has its own uncertainty as we do not know of any way to completely eradicate the uncertainty, we can find out a way of dealing with such uncertainties and the Gaussian method is a way to do so. In gaussian process, we describe the probability distribution over functions and then use Bayes rule to update our distribution of function using training data.

**Screen shot of the predicted versus actual visualisation of test data:**



**Statistics for the best performing model based on test data:**



**How would the optimum model change if transparency were very important:**

If transparency is the most important factor then we can use models which are based on glm and are tree based. Among our candidate models randomforest is transparent with only one tuning parameter and low rmse. GLMNet can also be used as it is transparent and does implicit feature selection and regularization. We can also use models like Lasso and ridge regression which does implicit feature selection and regularization and can be explained with the help of formula.

**Whether you recommend utilising an ensemble of the best n models?**

The motivation for using ensemble models is to reduce the [generalization error](https://www.sciencedirect.com/topics/computer-science/generalization-error) of the prediction. It combines several base models in order to produce one optimal predictive model. For getting ensemble model we may choose set of best models which are not statistically distinguishable. This can be seen with the help of the notches in boxplot. **The ensemble models can be considered as good option in this case if the rmse is decreasing further**. However, if transparency and interpretability is the focus and we have already got a model which can explain the dataset very closely, ensembling is not required as the model that is closest to the true data generating process will always be best and will beat most ensemble methods