Table of Contents

Introduction

The Shiny app is been provided to find the candidate variables and candidate models for the given dataset using the Caret package.

In the current report, I have focused on:

- Exploratory Data Analysis of the dataset
  - Summary
  - Missing data
  - Univariate analysis
  - Multivariate analysis
- Preprocessing steps
  - Data partitioning
  - Resampling
  - Dealing with missingness
  - Standardising
  - Adding and removing variables
  - Creating dummy variables
  - Dimension reduction
  - Feature filtering
- Methods
  - Method selection
  - List of methods
- Models
  - Train a model
  - Model evaluation and comparison
  - Visual summary of metrics
  - Choosing the best model
  - Performance on holdout data
  - Model-based outlier
  - Method description

- Transparent Model
- Conclusion

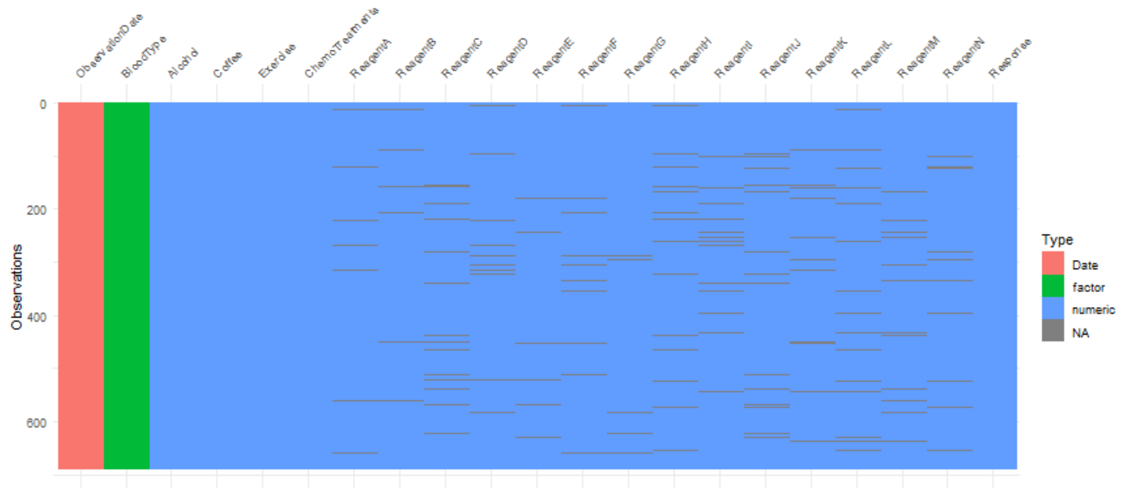
Exploratory Data Analysis of the dataset

Summary

The given dataset is a collection of 22 variables and 690 observations. Among these 22 variables, 19 variables are numeric, two categorical variables and one Date type. The Patient column that is an ID role has a high cardinality, which has the same number of rows, while reading the dataset in R it is converted into row names. The Response is a target variable that plays an outcome role. The Blood group is a nominal variable. The observationDate is converted to Date data type.

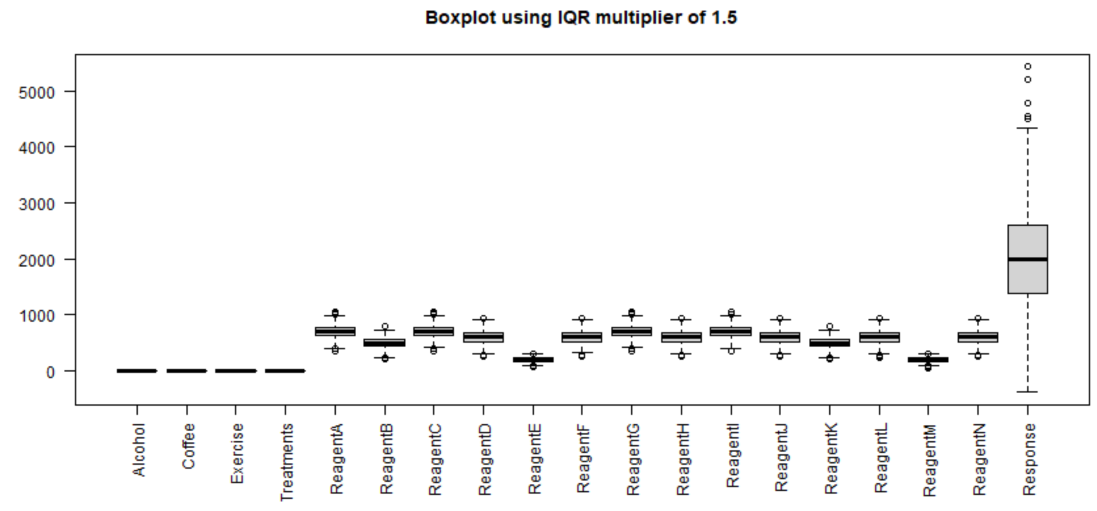
Missing data

The missing value chart below shows the number of missing values in each variable in a dataset. The ‘Patient’, ‘ObservationDate’, ‘Alcohol’, ‘Coffee’, ‘Exercise’, ‘ChemoTreatments’, ‘BloodType’ and ‘Response’ are the variables with no missing values. The numeric variables ‘ReagentA’ to ‘ReagentN’ are 14 variables that have reasonable missing values. The missing values appear to be random, as these values do not relate to any other data in the dataset and they have no discernable pattern.



Univariate analysis

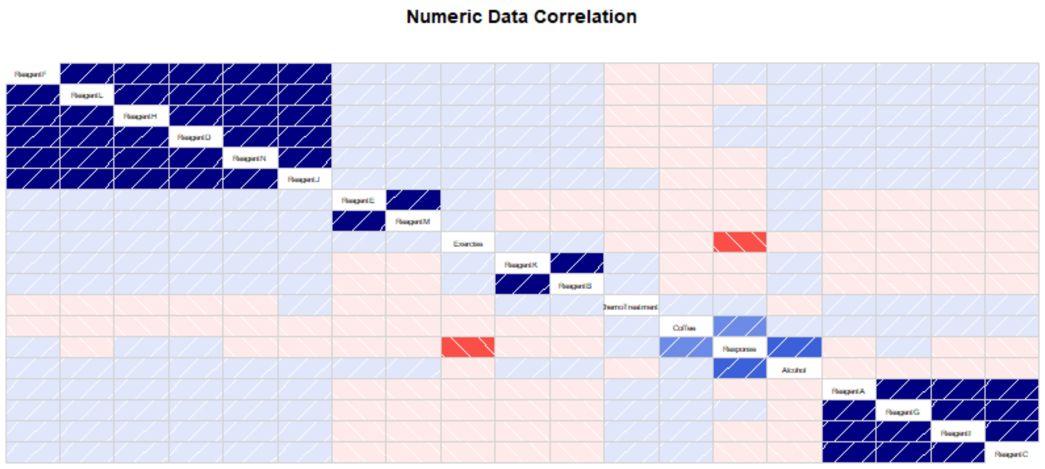
The Box plots help us to find the uni-variable outliers and skewness of the data. We have five different sets of variables that appear to be similar when the data is not standardised. These sets are ‘Alcohol’, ‘Coffee’, ‘Exercise’ and ‘Treatments’; ‘ReagentA’, ‘ReagentC’, ‘ReagentG’ and ’ReagentI’; ‘ReagentH’, ‘ReagentD’, ‘ReagentF’, ‘ReagentJ’, ‘ReagentL’ and ‘ReagentN’; ‘ReagentB’ and ‘ReagentK’; ‘ReagentE’ and ‘ReagentM’. The ‘ReagentA’ to ‘ReagentN’ have high and low outliers. Even when all these variables are scaled and centred, ‘ReagentA’ to ‘ReagentN’ variables have both high and low outliers. These outliers disappeared when *IQR is equal to 2, so these were not of great significance.



Multivariate analysis

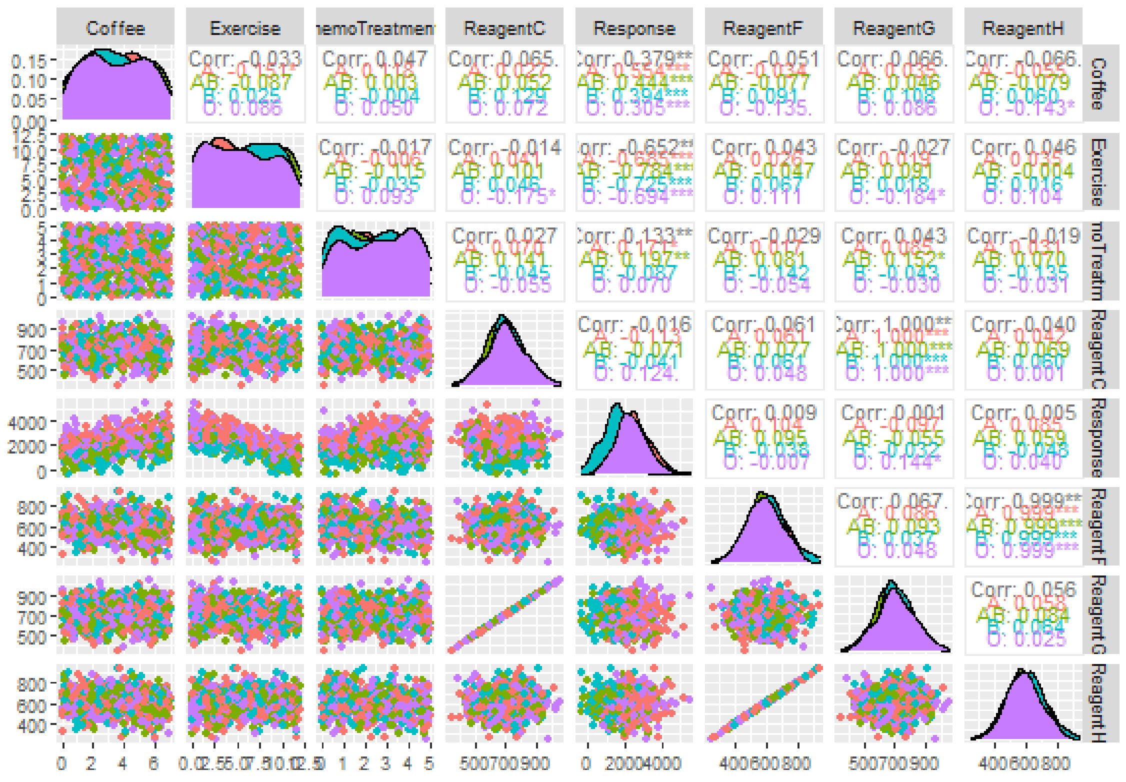
Correlation charts

Only numeric data is used to show the correlation between variables in the same chart. By just grouping the variables by OLO using the Pearson method, 5 different groups show high correlation: ‘ReagentA’, ‘ReagentC’, ‘ReagentG’ and ’ReagentI’; ‘ReagentH’, ‘ReagentD’, ‘ReagentF’, ‘ReagentJ’, ‘ReagentL’ and ‘ReagentN’; ‘ReagentB’ and ‘ReagentK’; ‘ReagentE’ and ‘ReagentM’. ‘Alcohol’, ‘Coffee’ and outcome variable ‘Response’ are correlated with each other. Whereas, ‘Exercise’ is negatively correlated with the outcome variable. Hence, as shown in the box plot, even here we can find five different groups of correlation between the numeric variables in the data set.



Pairs plot

The pairs plot used for this report is plotted in a separate Shiny app, to avoid making any changes to the structure of the provided app. Here the numeric variables were grouped by 'Bloodtype' to see the distributions. From the pairs plot, we can see that 'Alcohol', 'Coffee' and some 'Reagents' have no relationship with the 'Response' variable as we can see the points are scattered all over. 'Exercise' has a negative relationship with the 'Response' variable.



Pre-processing

Data partitioning

The createDataPartition() is used to split the data into unseen data and training data. Where the p parameter sets the percentage of the data that has training data. A slider control is used and by default, it is set to 80%. The remaining is the unseen data, which is used to predict. We get the data in matrix format by declaring list = FALSE.

All the other pre-processing steps are performed on training data only.

Resampling

Bootstrap sampling is the technique wherein with-replacement random sampling is used. In the original dataset, we have 690 observations; a bootstrap sample will also have 690 observations in it. After partitioning, the train data set has only 80% of the original data in it. To get the same size (690 observations) as the original dataset, bootstrap has to repeat this sampling many times. The process of repeating the samples several times is called as resampling.

The **trControl** parameter of caret::train(), is used to specify the type of resampling. When the caret :: train() is called the defult trControl is applied. In trControl, we have different arguments to specify like method, number, search type and setting the seed. Some of the arguments used in the code is as follows:

```
trainControl(method = "boot", number = 25, repeats = NA, allowParallel = TRUE, search = "grid",
            index = caret::createResample(y = y, times = n), savePredictions = "final", seeds = seeds,
            trim = TRUE)
```

For bootstrap resampling, the “boot” method is used along with 25 resamples, allowing us to process in parallel. The gridsearch is performed for optimizing the hyper-parameters, saving the outcome and then setting the seed.

Dealing with missingness

From the missingness plot, we had missing values in our data. Partial deletion is a technique where the missing values are removed. Some methods like generalised linear methods and support vector machine methods are not tolerant of missing values, therefore partial deletion is done using step_naomit(). Imputation is another way to deal with missingness. In the kNN imputation, the missing value for an observation is treated as a targeted response and is predicted on the mean if it is a numeric variable or mode if it is a categorical variable of the k nearest neighbours; the number of neighbours is set to 5. Bag imputation is also used. At first, all the candidate models were trained using kNN imputation as it is a more robust and sensitive method for missing values. The top three models (brnn, svmPoly, cubist) from the candidate models were slightly improved by using bag imputation. Comparing kNN imputation with bag imputation, the latter is very slow to train but had a slight improvement in their metrics.

Standardising

The data has to be scaled and centered using step_centre() and step_scale() as a steps in recipe(). step_center() will normalise numeric data to have a mean of zero. While step_scale() also normalises the numeric data to a standard deviation of one. For most of the candidate methods, the data is to be normalised since the numeric variables have different means and variances in them.

Adding and removing variable

Three new variables were created by splitting the 'observationDate' variable which is of Date data type. The three variables are day-of-the-week ('dow'), 'month' and 'decimal' where the date is converted into a decimal form of the year. The 'dow' variable is added and the 'observationDate' is removed while training the model. When the other new variables were added, there was a huge difference in the RMSE values obtained. By adding only the 'dow', the RMSE values were minimised.

Creating dummy variables

The ‘BloodType’ and ‘dow’ are low cardinality nominal variables that are converted to numeric variables by treating them with dummy encoding. For tree-based methods, dummy encoding is not used as a pre-processing step. If used, the sparse data will create dense data thereby losing the computational efficiency.

Dimension reduction

Dimension reduction is an approach to filter the non-informative features. As seen in the correlation chart, there were groups of numeric variables that are highly correlated. These variables can be eliminated by the Principal Component Analysis (pca) as a step_pca() in the recipe.

Feature filtering

The Zero variance variable that has a single unique value will not provide any useful information to the model. Hence, to filter these types of variables step_zv() is a step used in the recipe. Likewise, non-zero variables also provide less information to the model; step_nzv() step is used in the recipe to filter these types of variables. Other than these the methods like glmnet, rpart, glmStepAIC, gbm and M5 all have a characteristic of implicit feature selection. These methods were employed to reduce a large number of variables that were created after dummy encoding.

To achieve better results, centring and scaling are followed by kNN imputation in glmnet model, glmStepAIC model, gamSpline model and pls model. For the rest of the other candidate models, kNN imputation is followed by centring and scaling.

Methods

Method selection

Since the given dataset favours non-linear models, based on that the candidate methods were selected from each of the four classical flavours of the method. Like lm, glm, glmStepAIC and glmnet from Ordinary Least Square (OLS) methods; M5, cubist and gbm from tree-based methods; svmLinear, kNN and svmPoly from kernel methods and nnet and brnn from neural network methods. Along with these other wild card methods were also tried. Based on the regression map from the tutorial, if a method was performed well, methods similar to it and close to it from the map were chosen and tried. The pre-processing steps were tried in different combinations that suit the method best. The fine-tuning of the hyper-parameter like changing the tuneLength based on the method is tried. Some of the changes are discussed in the table.

List of methods

Model	Characteristics	Notes	Reason to choose
Generalised Linear Model (glmnet)	Generalised Linear Model Implicit Feature Selection L1 Regularisation L2 Regularisation Linear Classifier Linear Regression	2 hyper-parameters centring and scaling is done before imputation	OLS method that does feature selection and intolerant for missing values
Partial Least Squares (pls)	Partial Least Squares Feature Extraction Linear Classifier Linear Regression	1 hyper-parameter. The pre-processing step for transformation is added the RMSE value was increased	It creates new predictors based on the linear combinations of the original predictors which helps to identify any predictive relationship if exists
CART (Rpart)	Tree-Based Model Implicit Feature Selection	2 hyper-parameter doesn’t	Random example of a tree-based method, where it can return

	Handle Missing Predictor Data Accepts Case Weights	require any processing steps except adding variables in this case	only a subset of the features and it is tolerant to missing values
Linear Regression (lm)	Linear Regression Accepts Case Weights	1 hyper-parameter	A basic OLS method where we can find the relationship between the variables
Generalised Linear Model with Stepwise Feature Selection (glm-StepAIC)	Generalised Linear Model Feature Selection Wrapper Linear Classifier Implicit Feature Selection Two Class Only Accepts Case Weights	1 hyper-parameter Failed while missing is present	OLS method that does feature selection and intolerant for missing values can deal with binary class data
Stochastic Gradient Boosting (gbm)	Tree-Based Model Boosting Ensemble Model Implicit Feature Selection Accepts Case Weights	4 hyper-parameters Was too slow to train	A method from tree based method where it selects the important variables
Cubist	Rule-Based Model Boosting Ensemble Model Prototype Models Model Tree Linear Regression Implicit Feature Selection	2 hyper-parameters Failed when tried to add dummy variables	A tree-based method where the model rules are adjusted using the nearby points
Model tree (M5)	Rule-Based Model Tree-Based Model Linear Regression Implicit Feature Selection Model Tree	3 hyper-parameters due to the resampling each time it gives different results. Like getting a single node tree at times and when the method is trained again it	Tree-based model that predicts the outcome using linear model

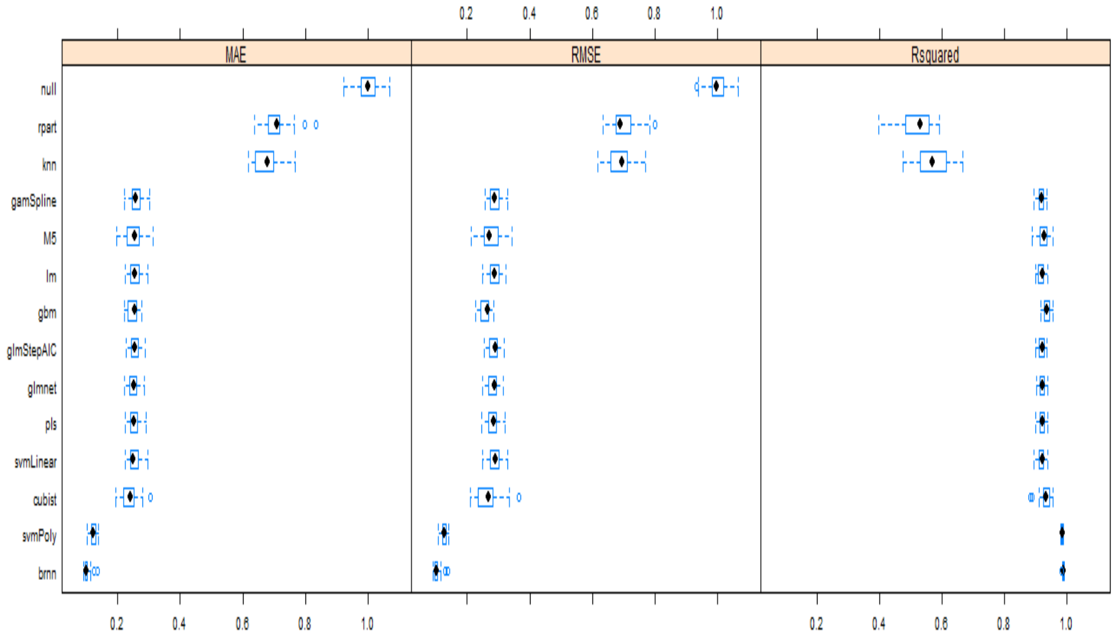
		will give an error	
Support Vector Machines with Linear Kernel (svmLinear)	Kernel Method Support Vector Machines Linear Regression Linear Classifier Robust Methods	1 hyper-parameter slow to train by applying pls due to correlation in data the RMSE values were high	Is a Kernel method works well when there is a clear margin of separation between classes
Support Vector Machines with Polynomial Kernel (svmPoly)	Kernel Method Support Vector Machines Polynomial Model Robust Methods	3 hyper-parameters slow to train. Tried with different tuneLength but got a minimum RMSE value when it is set to three	It is a kernel method that can handle non-linear classification. With more than two classes
k-Nearest Neighbors (kNN)	Prototype Models	1 hyper-parameter	Random example based on Kernel methods
Generalised Additive Model using Splines (gamSpline)	Generalised Linear Model Generalised Additive Model	1 hyper-parameter. Failed when missing values were present	It is a generalised method where it is not tolerant for missing values can deal with non-linear data
Bayesian Regularized Neural Networks (brnn)	Bayesian Model Neural Network Regularisation	1 hyper-parameter. Had to change trainControl() to set seed = NULL	Can be used for any size of data, predicts well for non-linear regression
Neural Network (nnet)	Neural Network L2 Regularization Accepts Case Weights	2 hyper-parameters. Had to change trainControl() to set seed = NULL	As it a Neural Network method and can deal with non-linearity

Models

Train a model

Applying all the recipe steps along with trainControl() in which the control parameters are defined are passed as arguments to train(). Some of the other arguments passed are data, method, tuneLength and the metric. The method name is defined for the method and the value specified for tuneLength is varied based on the method like svmPoly where the tuneLength is set to 3, rpart is set to 15, pls is set to 25 and for most of the other methods, tuneLength is set to 5. RMSE is used as the metric to compare the candidate models to select the best model. The train(), then performs a training task using resampling to optimise any hyper-parameters. Instead of tuning these hyper-parameters using tuneLength, if we know the specific hyper-parameters for the specific methods we can set these parameters to tuneGrid using expand.grid(). Either of these should be used. But in this app, the preferred option is to go with tuneLength, since the Caret will automatically set some reasonable values for these hyper-parameters. In order to compare these candidate models, the trainControl specification is kept consistent for all the candidate models that were trained.

Model evaluation and comparison



The above figure shows three sets of box plots one for each of the metrics MAE, RMSE and Rsquared that are obtained by training the model. The best model is chosen based on these metrics. The range of the RMSe between the best and worst-performing models is much larger. The brnn model has a minimum RMSE value and whereas nnet model has a maximum RMSE value both are of NeuralNetwork models. The nnet model whose RSME value is 2231 has performed worse than the null model and the MAE value is NA. An option is provided in the app to hide the models which perform worse than the null model. By selecting the check box nnet model is omitted from the list of candidate models. Most of the candidate model's RMSE values range from 230 to 269.

Visual summary of metrics

Table: The list of processing steps, hyper-parameters and metrics for each of the candidate models used.

Model	**Processing steps**	**Hyper-parameters**	**Resampled Performance**		
			RMSE	**R^2**	**MAE**
glmnet	center, scale, impute_knn, dow, dummy, nzv, naomit	alpha = 1.00 lambda = 2.62	264.22	0.92	183.82
Pls	dow, center, scale, impute_knn, dummy, zv	ncomp = 12.00	264.48	0.92	184.67
Rpart	Dow	Cp = 0.04	650.42	0.52	520.66
Lm	dow, impute_knn, center, scale, dummy, nzv	intercept = TRUE	268.37	0.92	188.27
glm-StepAIC	center, scale, impute_knn, dow, dummy, nzv, naomit	Parameter = none	266.62	0.92	186.71
Gbm	dow, impute_knn, center, scale, dummy, nzv	n.trees = 500, interaction.depth = 4, shrinkage = 0.10, n.minobsinnode = 10	239.77	0.94	183.31
Cubist	impute_bag, center, scale, dow	committees = 20, neighbours = 9	245.64	0.93	174.21
M5	dow, impute_knn	pruned = Yes, smoothed = Yes, rules = No	254.87	0.92	185.94
svmLinear	dow, impute_knn, center, scale, dummy, zv	c = 1.00	268.74	0.92	186.34
svmPoly	dow, impute_bag, center, scale, dummy, nzv	degree = 2, scale = 0.10, C = 0.50	116.13	0.98	89.52
kNN	impute_knn, center, scale, dow, dummy, nzv	k = 13	639.27	0.57	496.25

gamSpline	dow, center, scale, impute_bag, dummy, nzv, naomit	df = 1.00	268.11	0.92	188.59
Brnn	dow, impute_bag, center, scale, dummy, zv	neurons = 3	93	0.99	74.94
Nnet	dow, impute_knn, center, scale, dummy, zv	size = 1.00, decay = 0	2231	NA	2035.44

From the table above, the brnn model, svmPoly model and cubist model are the set of good models whose RMSE was lower when compared to other candidate models. And from the figure, all the three models demonstrate outliers which is a bad characteristic. This might be because of the data alone.

The good models were fine-tuned with the following preprocessing steps:

- Adding a nominal variable
- Bag imputation
- Standardising – centring and scaling
- Creating dummy variables for all the nominal variables
- Getting rid of zero variance

The best model is determined from this set of good models.

Choosing the best Model

The brnn has minimum values for both RMSE and MAE, and a maximum value of Rsquared when compared to other models.

Performance on unseen data

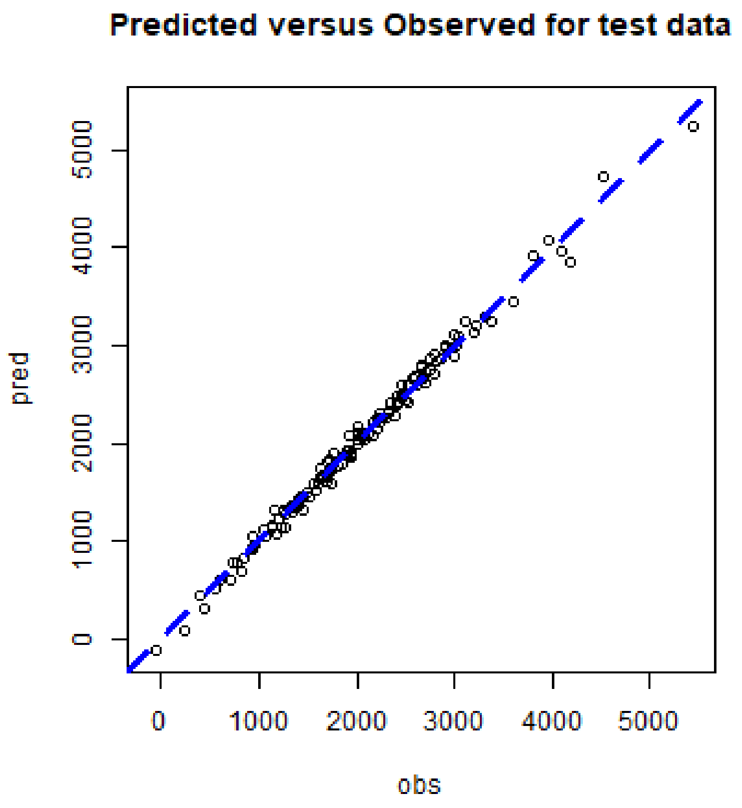


Figure: Predicted versus observed plot.

From the above figure, the points are all the test cases and the blue line is a line of perfect model. We have a perfect model and the scatter above the line represents the model accuracy.

Model-based outlier

The model-based outliers at IQR* 1.5 are shown below:

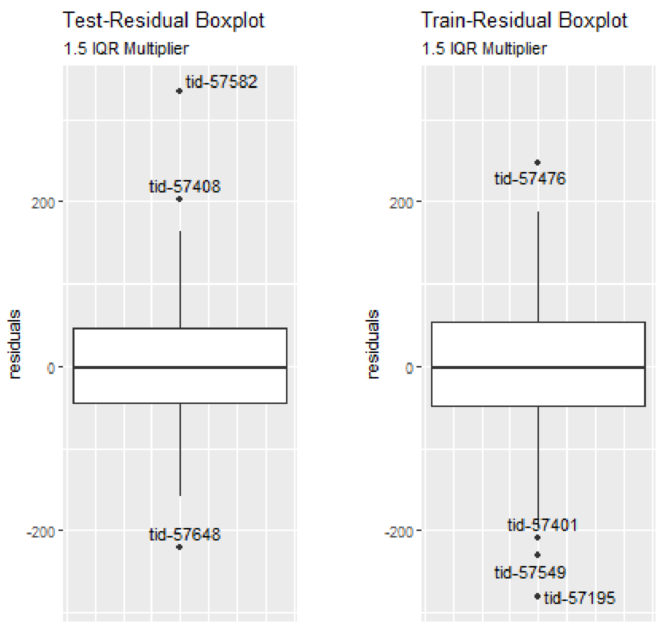


Figure: Test and Train residual box plots for IQR* 1.5.

And for IQR* 2.5 there is only one model-based outlier.

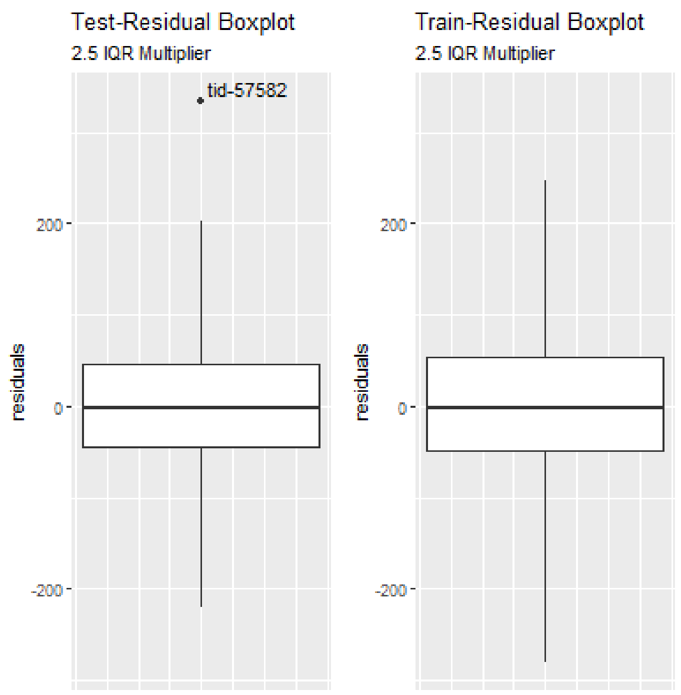


Figure: Test and Train residual box plots for IQR* 2.5.

The Patient Id tid-57582 might be a univariate outlier, even with an extreme IQR multiplier it falls outside the expected values.

Method description

The RMSE and MAE values are minimum and the R2 value is maximum for Baesian Reularisation Neural Network (brnn) model when compared with the metrics of other candidate models. The brnn is more robust than the back-propogation, it marginalise over the distribution of parameters in order to make predictions. It works well with all sizes of data. The brnn model might be the best model for the given patient dataset.

Table: Test metrics for brnn model.

Test Metric	**Value**
RMSE	93
MAE	74.94
R^2	0.99

Transparency

The best model brnn model cannot explain how the important predictors affect the outcome model, hence it is not considered as a transparent model.

The pls model provides better results when the data size is small. It is tolerant of missing values. It can deal with multicollinearity. The pls model was fine-tuned with the following preprocessing steps:

- Adding a nominal variable
- Standardising – centring and scaling
- kNN imputation where k = 5
- Creating dummy variables for all the nominal variables
- Getting rid of zero variance
- tuneLengh = 25.

Table: Test metrics for pls model.

Test Metric	Value
RMSE	264.48
MAE	184.67
R^2	0.92

From the candidate models, tree-based models and OLS based models are transparent. For the OLS based models, it is possible to extract and record the regression vector. Regression vector is a global thing that does not change from observation to observation. This can be used for global interpretation. The regression vector from the pls model is created by standardising all the predictors hence we have all the coefficients on the same scale.

Table: Showing the predictors and their coefficients obtained by the trained pls model.

Predictors	Coefficients
Alcohol	352.03
Coffee	314.21
Exercise	-557.52
ChemoTreatments	76.68
ReagentA	-109.19
ReagentB	-17.57
ReagentC	-32.85
ReagentD	5.14
ReagentE	47.68
ReagentF	40.89
ReagentG	99.36
ReagentH	73.94
ReagentI	0.79
ReagentJ	-9.30
ReagentK	-8.48
ReagentL	-36.69
ReagentM	-36.99
ReagentN	-9.61
BloodType_AB	-735.30
BloodType_B	-1079.48
BloodType_O	-287.85
ObservationDate_dow_Mon	174.32
ObservationDate_dow_Tue	-5.02
ObservationDate_dow_Wed	-79.48

ObservationDate_dow_Thu	48.32
ObservationDate_dow_Fri	-198.07
ObservationDate_dow_Sat	3.84

From the table, we can see that the variable ObservationDate_dow_Mon has a positive coefficient and it's one of the most important variables. When these coefficients were sorted in order we can see that ObservationDate_dow_Mon is the third most important variable followed by 'ChemoTreatments'. The first two being the 'Alcohol' and 'Coffee'.

Conclusion

The model could allow us to make some decisions based on the patients that were mostly diagnosed on Monday when compared to other days in a week. As all the results were depending on the day_of_week (dow) variable which is obtained by splitting observationDate. The quality of the diagnostics or the specialists who were diagnosing on Monday might also be the reason for getting these results. Or it might also be that Sunday being the weekend there were no results for that particular day. The weekend off for the consultants or the specialists might also be a factor for better results on Monday compared to other days in a week. There might be some other method that could be used in preference to brnn method where we can understand the reason why the patients were diagnosed only on Monday. So we cannot say that brnn is the best model for the given patient dataset.

References

Bayesian Regularized Neural Networks for Small n Big p Data. https://www.intechopen.com/chapters/50570.

Churning with Caret: non-linear models. http://philipmgoddard.com/modeling/nonlinearChur.