DATA423
Data Science for Industry

Assignment 3

Student Name: Wijesingha Mayadunnage Uditha Iresh Mayadunna

Student ID: 39768450

1 Contents

2 Table of Figures

3 Tables

4 Data Description

```
'data.frame':   969 obs. of  21 variables:
 $ Alcohol        : num  3.75 3.876 0.445 2.368 0.201 ...
 $ Coffee         : num  5.001 0.201 1.864 5.194 6.674 ...
 $ Exercise       : num  6.468 4.913 6.377 0.191 10.823 ...
 $ ChemoTreatments: num  2.15 3.2 2.82 3.22 1.12 ...
 $ BloodType      : Factor w/ 4 levels "A","AB","B","O": 4 3 2 1 1 3 1 3 1 3 ...
 $ ReagentA       : num  603 531 992 624 718 ...
 $ ReagentB       : num  323 566 519 645 514 ...
 $ ReagentC       : num  603 529 993 621 714 ...
 $ ReagentD       : num  541 658 NA 461 433 ...
 $ ReagentE       : num  200 249 218 203 173 ...
 $ ReagentF       : num  543 658 713 466 429 ...
 $ ReagentG       : num  602 528 993 621 713 ...
 $ ReagentH       : num  538 659 714 461 429 ...
 $ ReagentI       : num  602 525 NA 624 712 ...
 $ ReagentJ       : num  539 659 NA 462 426 ...
 $ ReagentK       : num  321 566 522 647 512 ...
 $ ReagentL       : num  545 664 707 461 430 ...
 $ ReagentM       : num  202 252 207 208 181 ...
 $ ReagentN       : num  541 664 714 466 432 ...
 $ ObservationDate: Date, format: "2013-09-18" "2013-09-20" "2013-09-21" "2013-09-23" ...
 $ Response       : num  3585 3059 1814 5281 1953 ...
```

Figure 1: Assignment 3 Data Summary

The Assignment 3 data set contains 969 entries across 21 different variables. Among these, 19 variables are numeric, one is a categorical variable, and one is a date variable. The date follows the format “YYYY-MM-DD”, and there is a Response variable as well. The "Blood Type" variable is a nominal variable with the cardinality of 4: A, B, AB, and O.

4.1 Missing Data

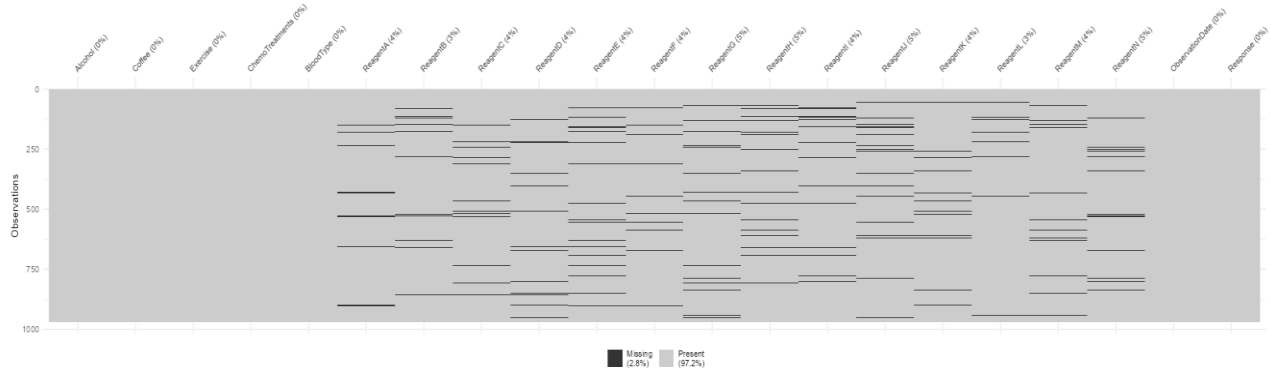

Figure 2: Vis Mis Plot

The vis miss plot in figure 2 indicates that a mere 2.8% of the overall data is missing, and no individual variable has more than 5% missing values.

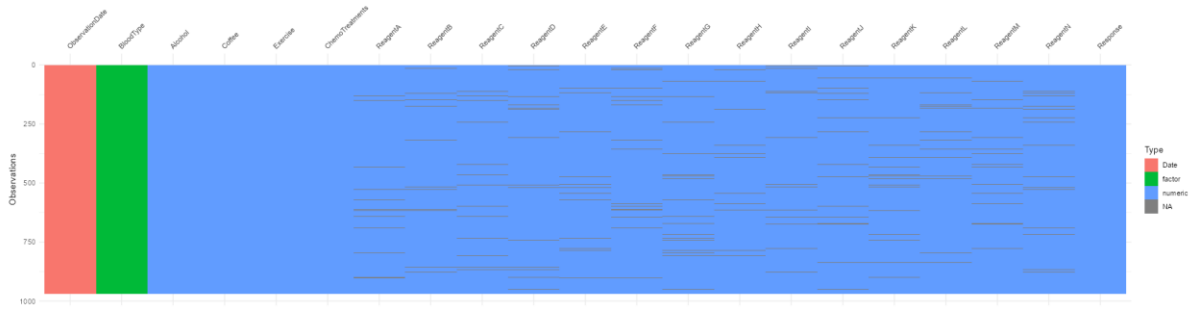

Figure 3: Vis_dat Plot

The vis dat plot in figure 3 reveals that only the numeric variables in the dataset contain missing data.

4.2 Outliers

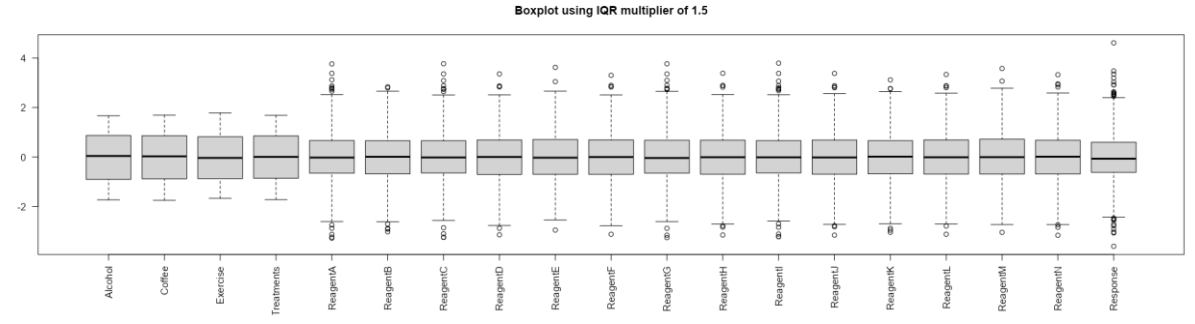

Figure 4: Boxplot Using IQR of 1.5

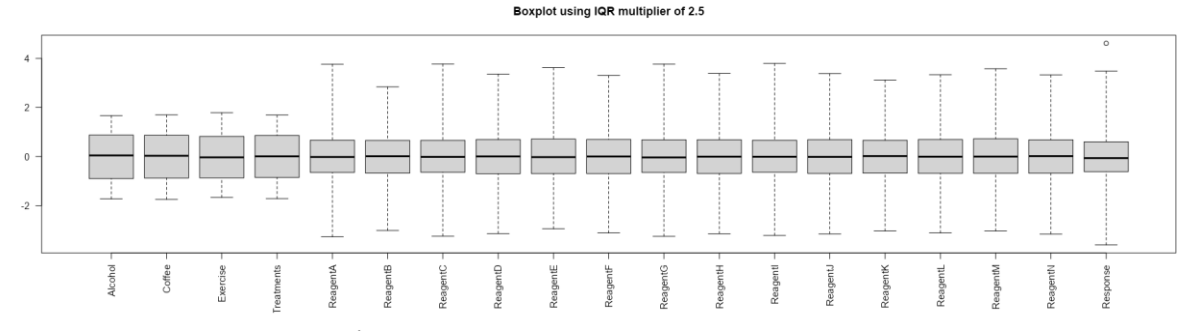

Figure 5: Boxplot Using IQR of 2.5

As shown in figure 4 and 5, when the Interquartile Range (IQR) multiplier is set to 1.5, outliers are present in all of the Reagent variables and in the Response variable. However, when the IQR multiplier is increased to 2.5, most of the outliers disappear, except for one outlier in the Response variable.

4.3 Correlation between variables

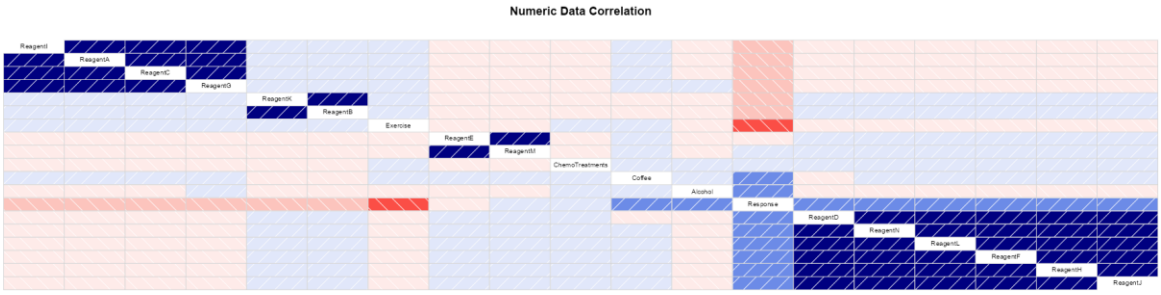

Figure 6: Numeric Data Correlation

The above correlation plot indicates that the Reagent variables D, N, L, H, F, and J are highly positively correlated. Similarly, the Reagent variables I, A, C, and G are also highly positively correlated. In contrast, the Exercise and Response variables exhibit a strong negative correlation.

5 Strategies

5.1 Missing data

Figures 2 and 3 demonstrate that there is no excessive missingness among both variables and observations. Additionally, the Rpart tree depicted in Figure 7 indicates that the missing values appear randomly distributed, showing no recognized patterns in the data's missingness.

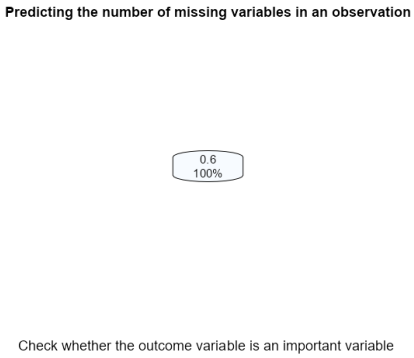

Figure 7: Rpart tree for missing data

5.2 Outliers

Figures 4 and 5's boxplots indicate that some variables have outliers. Nevertheless, most outliers become insignificant when an interquartile range (IQR) factor of 2.5 is used, avoiding the need to remove them. To reduce the influence of these data points, transformation preprocessing techniques can be used rather than eliminating them. Using techniques and algorithms that are naturally resistant to outliers, such tree-based methods, can also be helpful. This approach maintains the accuracy and adaptability of the model while allowing for the retention of all data points.

6 Processing

6.1 Data Sampling

Out of a total of 969 observations, 777 were allocated for training (80%) and 192 were set aside for testing (20%).

6.2 Hyper-Parameter fine tuning

The hyper-parameters were fine-tuned using 25 bootstrap re-samples drawn from the training data. This approach ensured that the model's performance assessment and parameter optimization were both robust and reflective of diverse scenarios within the training dataset. Also, it enables parallel processing to enhance computational efficiency, and performs a grid search on hyperparameters.

6.3 Pre-Processing Steps

The following preprocessing techniques were used to optimize the resampling metrics and the sequence was decided according to the model or the method.

6.3.1 Imputation of Missing Values

6.3.1.1 impute_knn:

The K-nearest neighbours algorithm imputes missing values by estimating them based on the similarity between observations, using the closest neighbours' data.

6.3.1.2 impute_bag:

Bagging, or Bootstrap Aggregating, imputes missing values by building multiple models from different data subsets and averaging the results.

6.3.1.3 impute_median:

Imputation of missing values by replacing them with the median of the non-missing values in each feature.

6.3.1.4 impute_mode:

Imputation of missing values by replacing them with the mode (most frequent value) of the non-missing values in each feature.

6.3.2 Partial Deletion

6.3.2.1 naomit:

Removes observations (rows) that contain any missing values.

6.3.3 Centering and Scaling

6.3.3.1 center:

Centering data involves subtracting the mean value from each feature, resulting in each feature having a mean of zero.

6.3.4 scale:

Scaling the data involves dividing each feature by its standard deviation, resulting in each feature having a standard deviation of one.

6.3.5 Handling Date Features

6.3.5.1 month:

Extracts the month from date-type features and converts it into a factor variable.

6.3.5.2 dow:

Extracts the day of the week from date-type features and converts it into a factor variable.

6.3.5.3 dateDecimal:

Converts date-type features into a decimal representation of the year.

6.3.6 Transformation

6.3.6.1 YeoJohnson:

The Yeo-Johnson power transformation normalizes data by adjusting both positive and negative values, functioning similarly to the Box-Cox transformation.

6.3.6.2 pca:

Principal Component Analysis (PCA) transforms data by reducing its dimensionality, projecting it onto principal components that capture the most variance.

6.3.6.3 pls:

Partial Least Squares (PLS) transformation is similar to PCA but additionally considers the target variable when constructing the components to better capture relevant variance.

6.3.6.4 ica:

Independent Component Analysis (ICA) transformation separates data into independent components, commonly used in signal processing to identify underlying factors or sources.

6.3.7 Variable Selection and Reduction

6.3.7.1 nzv:

Removing near-zero variance predictors helps eliminate features with very low variance that typically offer minimal useful information for modelling.

6.3.7.2 zv:

Removing zero variance predictors eliminates features that provide no useful information for modelling, as they do not vary at across observations.

6.3.7.3 corr:

Removing highly correlated predictors eliminates redundant features that share a high degree of correlation with others, streamlining the modelling process.

6.3.8 Feature Engineering

6.3.8.1 dummy:

Creating dummy variables from categorical features involves converting each level of a categorical variable into a separate binary feature, facilitating the use of these categories in numerical analysis models.

6.3.8.2 poly:

Creating polynomial terms for numeric features allows for the modeling of non-linear relationships by generating higher degree variables, thereby enhancing the ability to capture complex patterns in the data.

6.3.8.3 interact:

Creating interaction terms between features captures dependencies between multiple variables, allowing models to better understand the combined effects of these variables on the outcome.

6.3.9 Indicator and Flagging

6.3.9.1 indicate_na:

Adding binary indicators for missing values creates a binary feature for each column to indicate the presence of missing values, enhancing data analysis by explicitly marking where data is incomplet.

6.4 Preprocessing order Example

Initially, one preprocessing technique from each category was selected to identify candidate models. Subsequently, additional preprocessing methods were explored to further optimize these models. Here is an example steps for preprocessing.

1. Imputation – impute_knn
2. Centering and Scaling – Centre, Scaling
3. Handling Date Features – dow
4. Tranformation - YeoJohnson
5. Variable Selection and Reduction – corr

7 Methods

The methods evaluated were grouped into five primary categories: Ordinary Least Squares (OLS), Tree-Based, Kernel-Based, Neural Networks (NN), and Ensemble with all remaining techniques classified as "Other." In all, 32 techniques from these categories were tried to establish acceptable candidate models for each group, with 6 failing to train. Here are the specifics of these trials.

7.1 Trials

Method	Characteristics	Notes	Reason chosen
NULL		Quick to train Trained successfully	Already Chosen
GLMnet	Generalized Linear Model Implicit Feature Selection	2 hyperparameters Quick to train Trained successfully	Already Chosen
PLS	Partial Least Squares Feature Extraction	1 hyperparameter Quick to train Trained successfully	Already Chosen
Rpart	Tree-Based Model Implicit Feature Selection Handle Missing Predictor Data Accepts Case Weights	1 hyperparameter Quick to train Trained successfully	Already Chosen
Random Forest	Ensemble Leaning Method Bagging	1 hyperparameter Slow to train Trained successfully	Random Selection
GBM	Tree-Based Model Boosting Ensemble Model Implicit Feature Selection Accepts Case Weights	4 hyperparameter Took moderate time to train Trained successfully	Random Selection of a tree-based model which has Ensemble characteristics
glmboost	Generalized Linear Model	2 hyperparameter	Random Selection of an

	Ensemble Model Boosting Linear Classifier	Trained successfully	OLS model which has Ensemble characteristics
NNSL	Linear Regression	Trained successfully	
SVM Radial	Kernel Method Support Vector Machines Radial Basis Function	2 hyperparameter Trained successfully Slow to train Used parallel processing	Example for a kernel method
Robust Linear	Linear Regression Robust Model Accepts Case Weights	2 hyperparameter Took moderate time to train Trained successfully	Another OLS methos
Rpart SE	Tree-Based Model Implicit Feature Selection Handle Missing Predictor Data Accepts Case Weights	Quick to train Trained successfully	Another Rpart method
Rpart 2	Tree-Based Model, Implicit Feature Selection, Handle Missing Predictor Data Accepts Case Weights	1 hyperparameter Quick to train Trained successfully	Another version of Rpart
Conditional Inference Tree	Tree-Based Model Implicit Feature Selection Accepts Case Weights	1 hyperparameter Trained successfully	Random Selection of a tree-based method
Neural Network	Neural Network	Trained successfully	A basic neural network method
pcaNNet		Trained successfully	Random selection of a Neural Network Method
avNNet	Neural Network Ensemble Model Bagging L2 Regularization	3 hyperparameter Trained successfully	Random selection of a Neural Network Method which has Ensemble

	Accepts Case Weights		characteristics.
monmlp	Neural Network	2 hyperparameter Trained successfully	Random selection of a Neural Network Method
Black Boost	Tree-Based Model Ensemble Model Boosting Accepts Case Weights	2 hyperparameter Trained successfully	Random Selection of a tree-based method which has Ensemble characteristics
Boosted Trees	Tree-Based Model Ensemble Model Boosting	3 hyperparameter Trained successfully	Random Selection of a tree-based method which has Ensemble characteristics
Rvm Radial	Kernel Method Relevance Vector Machines Radial Basis Function Robust Methods	1 hyperparameter Trained successfully	Random Selection of a kernel method
SVM Radial Cost	Kernel Method Support Vector Machines Radial Basis Function	1 hyperparameter Trained successfully	Another SMV Radial method
kNN	Prototype Models	1 hyperparameter Trained successfully	
Gausspr Radial	Kernel Method Gaussian Process Radial Basis Function	1 hyperparameter Trained successfully	Random Selection of a kernel method
QRNN	Neural Network L2 Regularization Quantile Regression Bagging Ensemble Model Robust Model	3 hyperparameter Error occurred while training Failed to train	Random selection of a Neural Network Method which has Ensemble characteristics
M5	Rule-Based Model	3 hyperparameter	Random Selection of a

	Tree-Based Model Linear Regression Implicit Feature Selection Model Tree	Trained successfully	tree-based method
M5Rules	Rule-Based Model Linear Regression Implicit Feature Selection Model Tree	2 hyperparameter Trained successfully	Random Selection of a tree-based method
Gaussian Process with Polynomial Kernal	Kernel Method Gaussian Process Polynomial Model	2 hyperparameter Error occurred while training. Training failed	Random Selection of a polynomial kernel method
Bagged Trees	Tree-Based Model Ensemble Model Bagging Accepts Case Weights	Trained successfully	Random Selection of a tree-based method which has Ensemble characteristics
Bayesian regularized Neural Network	Bayesian Model Neural Network Regularization	1 hyperparameter Keep training the model for a long time. No result. Training failed	Random selection of a Neural Network Method
Kernal Regularized least Squares (Polynomial)	Kernel Method L2 Regularization Polynomial Model	2 hyperparameter Slow to train Trained successfully	Random Selection of a polynomial kernel method
Multi-Layer Perceptron	Neural Network	1 hyperparameter Trained successfully	Random selection of a Neural Network Method
SVM with Polynomial Kernal	Kernel Method Support Vector Machines Polynomial Model Robust Methods	3 hyperparameter Keep training the model for a long time. No result. Training failed	Random Selection of a polynomial kernel method
Extreme	Neural Network	2 hyperparameter	Random selection

Learning Machine(elm)		Error in installing the package “elmNN” Training failed	of a Neural Network Method
lmStepAIC	Linear Regression Feature Selection Wrapper Accepts Case Weights	Fast to train Trained Successfully	Random selection of an OLS method
mxnet	Neural Network	Error in installing the package “maxnet” Training failed	Random selection of a Neural Network Method
leapSeq	Linear Regression Feature Selection Wrapper	1 hyperparameter Fast to train Trained Successfully	Random selection of a Neural Network Method

8 Candidate Models

During the trials, various preprocessing techniques were applied to enhance the performance of each method. The top methods from each category that performed well were then chosen as candidate models.

8.1 Ordinary Least Squares (OLS) Models

For the Ordinary Least Squares (OLS) methods, four different approaches were tried, and the optimum results for each method are summarized as follows (Null Model is shown as the reference).

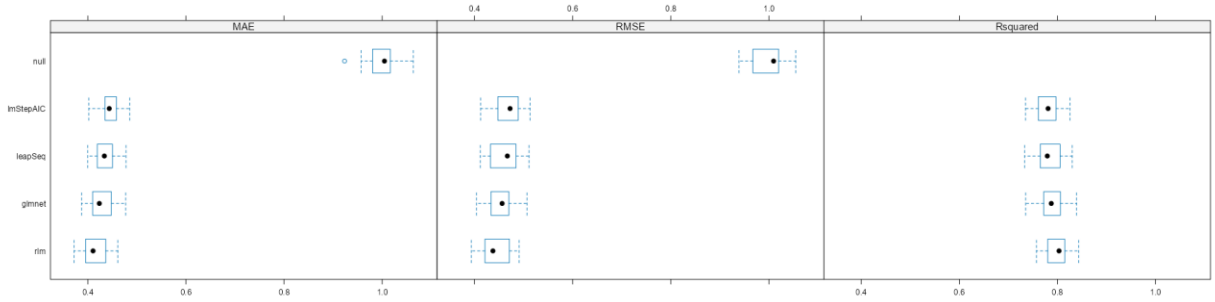

Figure 8: Resampled Performance of OLS models

Model	Processing steps	Hyper-parameters		Resampled Performance		
				RMSE	R2	MAE
rlm	Na omit Dow Dummy corr	Intercept psi	TRUE Psi.hampel	462.85	0.80	330.11
glmnet	Na omit YeoJohnson Dummy	Alpha Lambda	0.32 11.71	472.59	0.79	340.31
leapSeq	Na omit YeoJohnson month Dummy	nvmax	10	479.98	0.78	347.49
lmStepAIC	Na omit YeoJohnson Dummy			485.88	0.78	352.48

Table 1:Resampled Performance of OLS models

Based on the performance of the resamples, the robust linear regression (rlm) methods outperformed all others. The preprocessing steps for the rlm method begin with partial deletion, which is used to handle missing values. Following this, any date information is extracted and converted into a factor variable to facilitate categorical processing. The next

step involves applying dummy coding to convert categorical variables into a format suitable for modelling. Finally, the process concludes with the removal of highly correlated predictors, which helps to reduce multicollinearity and improve model performance.

8.1.1 Performance on test data

Similarly, with the resampled data, the robust linear regression (rlm) method demonstrated better performance on unseen data compared to other ols methods.

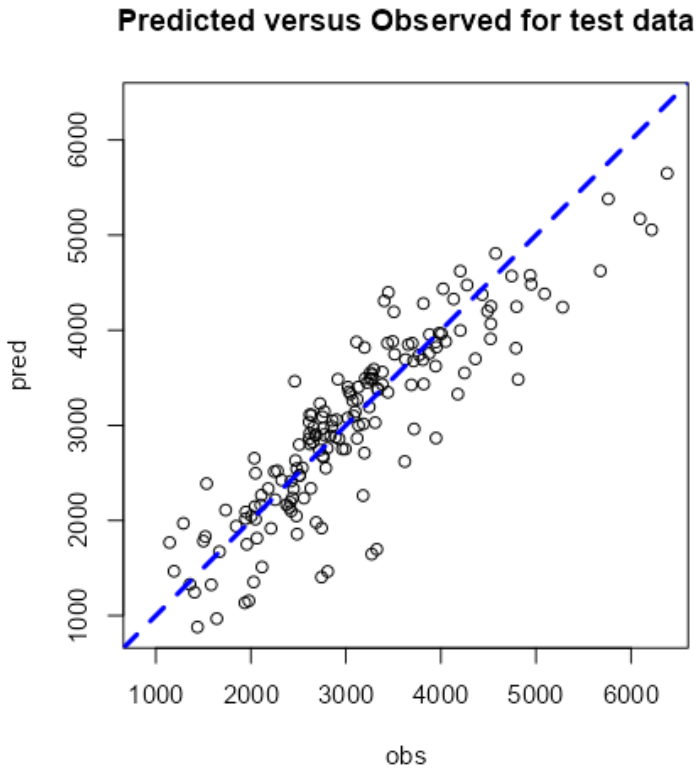

Figure 9: Predicted Vs Observed on Unseen Data - rlm model

Test Metric	Value
RMSE	493.477
MAE	364.179
R2	0.772

Table 2: Performance on Unseen Data - rlm model

The table 2 shows the test results of the rlm model and the RSME value 493.477 is higher than the Resamples RSME.

8.1.2 Residual Boxplots

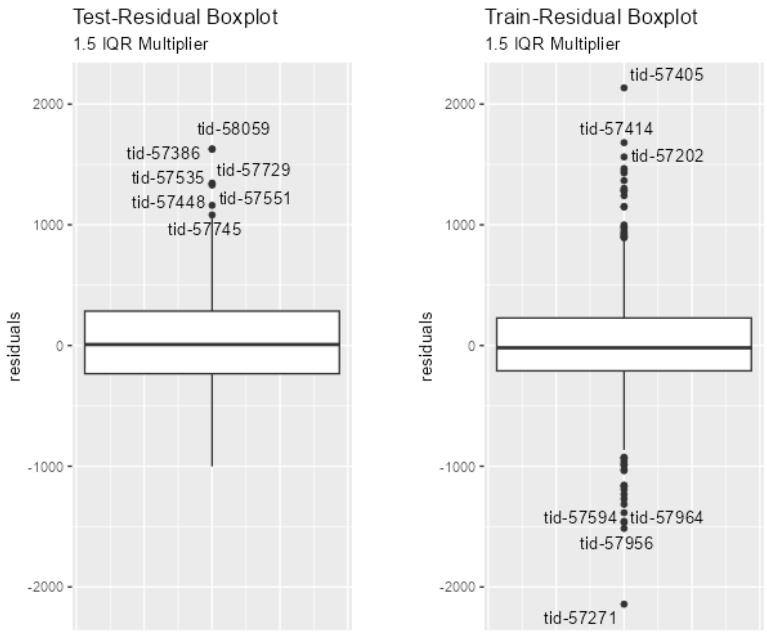

Figure 10: Residual Box Plots of rlm model

The residual boxplots in figure 10 indicate that there are more test residual outliers than train residual outliers when the interquartile range (IQR) is set to 1.5.

8.2 Tree Based Methods

There were 6 tree-based method trials and methods with lowest resampled RMSE values were selected as candidate models. The figure 11 shows the MAE, RMSE and Rsquared values for each tree-based methods and the null method is shown as the reference for all the others.

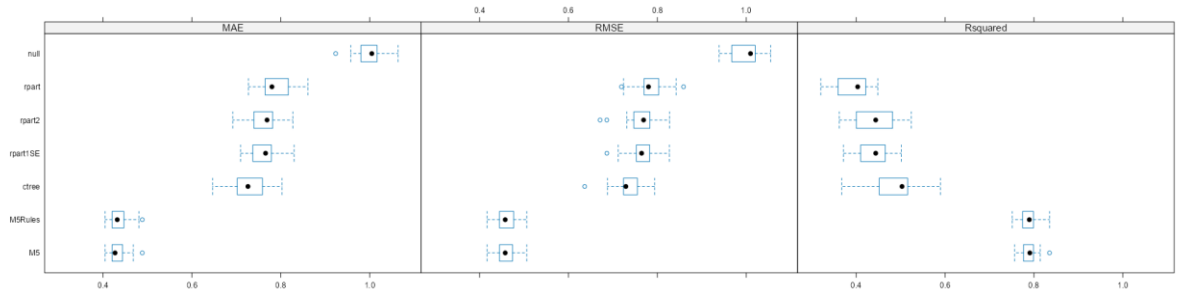

Figure 11: Resampled Performance of tree-based models

Model	Processing steps	Hyper-parameters		Resampled Performance		
				RMSE	R2	MAE
M5	Impute_knn Dow	pruned	yes	478.31	0.79	344.85
		Smoothed	yes			

	month	rules	no			
M5Rules	Impute_knn Dow month	pruned	yes	480.24	0.79	346.23
		Smoothed	yes			
ctree	Dow Dummy	Mincriterion	0.50	762.52	0.49	581.18
Rpart1SE	Dow month			792.76	0.44	607.16
Rpart2	Impute_median YeoJohnson Dow Month	maxdepth	5	792.02	0.44	606.87

Table 3: Resampled Performance of Tree-based models

Out of these models, M5 model performs better with the resampled data.

The preprocessing steps for the M5 method begin with knn imputation, which is used to handle missing values. Following this, any date information is extracted and converted into a factor variable to facilitate categorical processing. Finally the month is extracted and converted into a factor variable.

8.2.1 Performance on test data

Likewise, the resampled data, M5 model perform well on unseen data compared to other tree-based models.

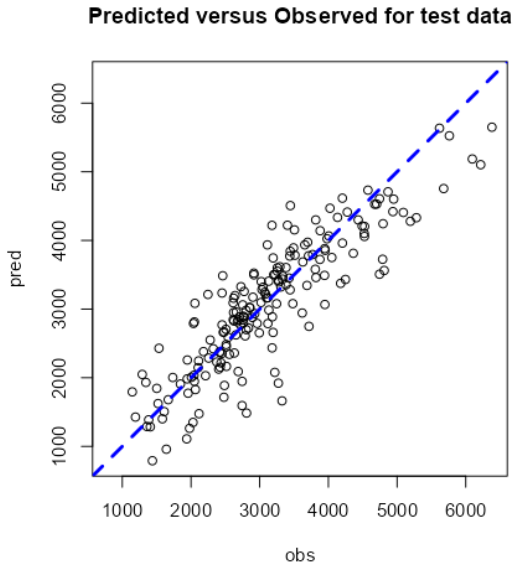

Figure 12: Predicted Vs Observed on Unseen Data - M5 Model

Test Metric	Value
RMSE	505.156
MAE	0.769
R2	378.269

Table 4: Performance on Unseen Data – M5 Model

Table 4 presents the test results for the M5 model, indicating an RSME value of 505.156. This value is higher than the Resampled RSME value recorded for the M5 method.

8.2.2 Residual Boxplots

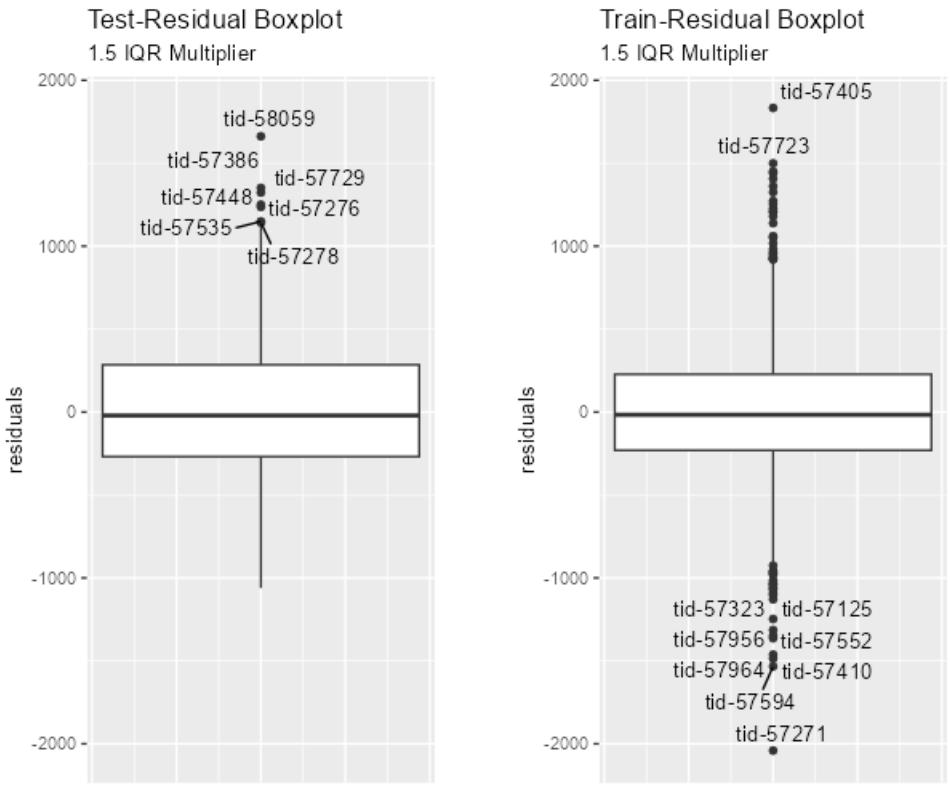

Figure 13: Residual Box Plots of M5 model

According to the residual boxplots in figure 13, the M5 model displays 7 test residuals and 10 train residuals when the interquartile range (IQR) value is set at 1.5.

8.3 Neural Network Models

Figure 14 and Table 5 display the optimized resample performance of each of the four neural network model trials.

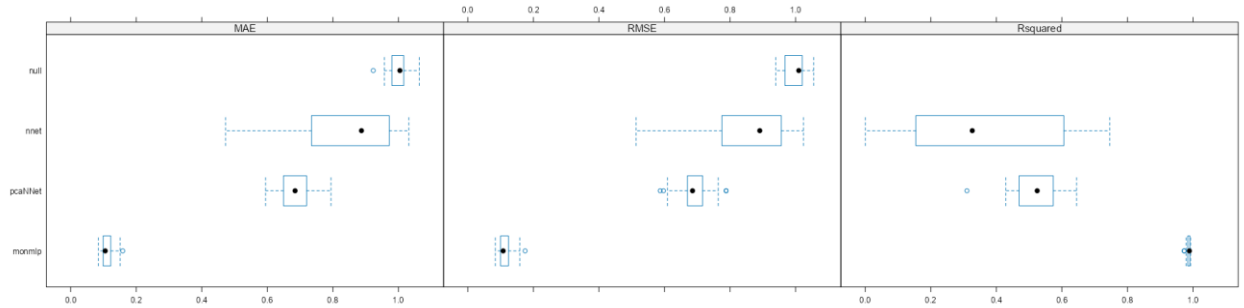

Figure 14: Resampled Performance of Neural Network models

Model	Processing steps	Hyper-parameters		Resampled Performance		
				RMSE	R2	MAE
monmlp	Impute bag Dow Dummy corr	Hidden1	5	118.50	0.99	89.53
		n.ensemble	1			
pcaNNet	Impute bag Dow Dummy corr	Size	5	718.08	0.53	549.79
		Decay	0.10			
nnet	Impute Knn Dow dummy	Size	5	878.19	0.36	664.20
		Decay	0.1			
mpl	Impute Knn Dow dummy	Size	3	1171.60	0.01	920.54

Table 5: Resampled Performance of Neural Network models

Out of all neural network models, monmlp model performs better with the resampled data.

The preprocessing steps for the monmpl method begin with bag imputation, which is used to handle missing values. Following this, any date information is extracted and converted into a factor variable to facilitate categorical processing. The next step involves applying dummy coding to convert categorical variables into a format suitable for modelling. Finally, the process concludes with the removal of highly correlated predictors, which helps to reduce multicollinearity and improve model performance.

8.3.1 Performance on test data

Likewise, the resampled data, monmlp model perform well on unseen data compared to other tree-based models.

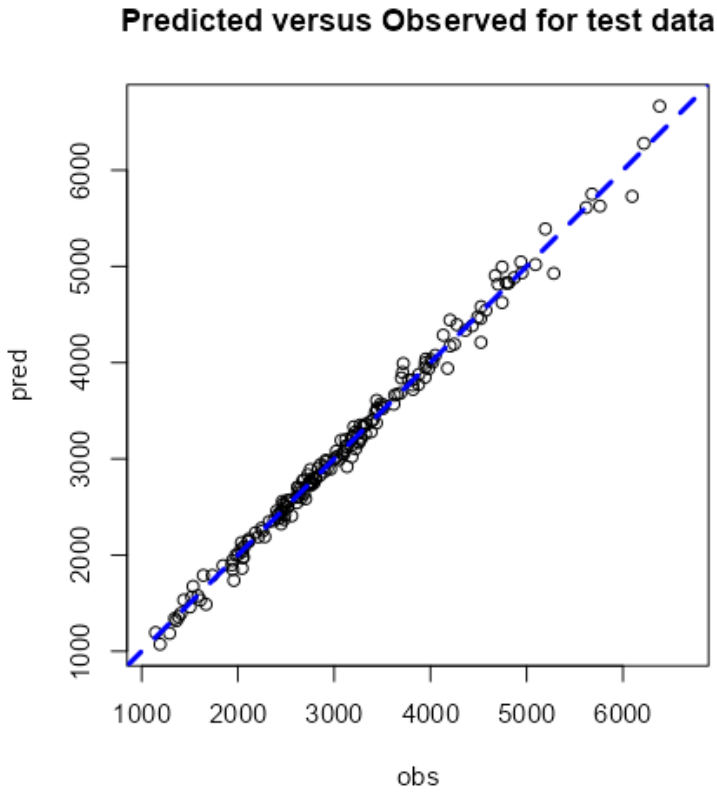

Figure 15: Predicted Vs Observed on Unseen Data - monmpl model

Test Metric	Value
RMSE	94.978
MAE	67.221
R2	0.992

Table 6: Performance on Unseen Data - monmpl model

Table 6 presents the test performance of the monmpl model, indicating that the RMSE value is 94.978, which is lower than the RMSE observed in the resampling phase. This suggests that the model may perform better on unseen test data compared to how it performed on the resampled data during validation, highlighting its potential effectiveness and generalization capabilities.

8.3.2 Residual Boxplots

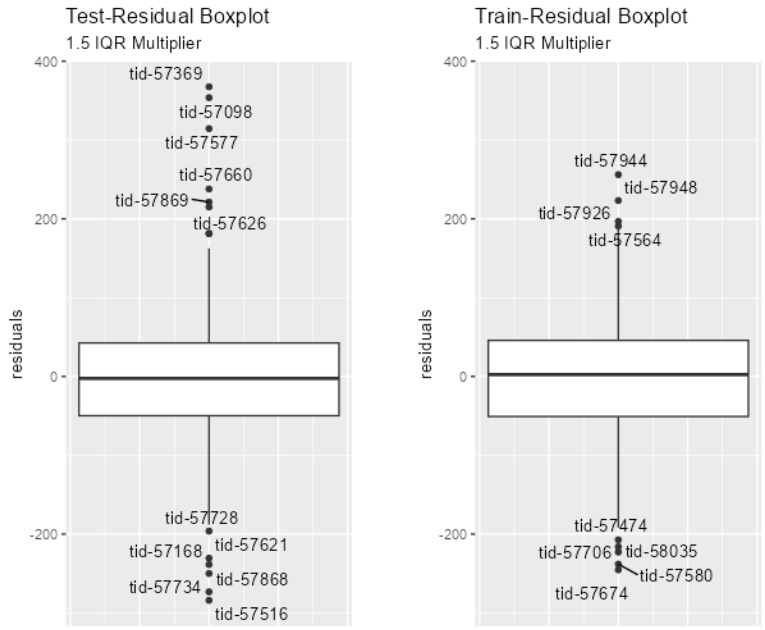

Figure 16: Residual Box Plots of monmpl model

The residual box plot in figure 16 indicates that there are more residual outliers in the test predictions compared to the train predictions.

8.4 Kernal Models

Figure 17 and Table 7 present the resampled performance of five different kernel method trials.

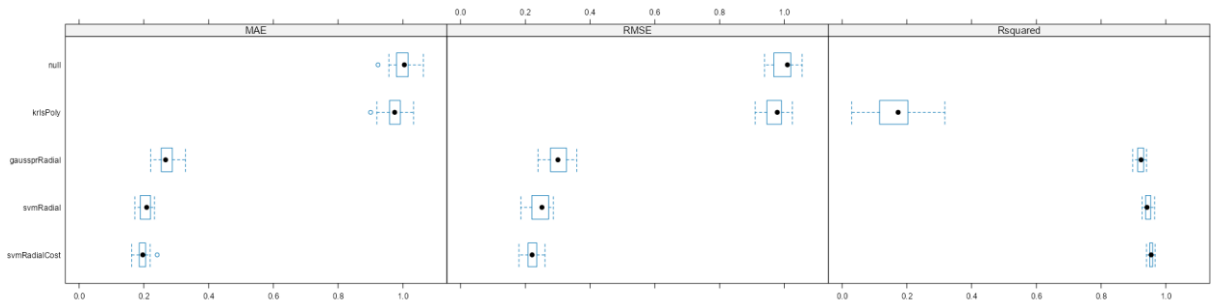

Figure 17:Resampled Performance of Kernal models

Model	Processing steps	Hyper-parameters		Resampled Performance		
				RMSE	R2	MAE
svmRadialCost	Impute knn Dow Dummy Corr	c	4	228.94	0.95	156.06
svmRadial	Impute knn Dow Dummy	Sigma	0.02	253.13	0.94	164.48
		c	16			
gaussprRadial	Impute knn Dow Dummy Corr	Sigma	0.04	313.83	0.92	214.42
krlsPoly	Impute knn Dow Dummy	Lambda	NA	1006.47	0.16	775.34
		degree	1			
rvmRadial	Naomit yeoJohnson dow dummy	sigma	0.02	3224.66	0.00	3060.99

Table 7: Resampled Performance of Kernal models

Out of these models, svmRadialCost model performs better with the resampled data. The preprocessing steps for the svmRadialCost method begin with knn imputation, which is used to handle missing values. Following this, any date information is extracted and converted into a factor variable to facilitate categorical processing. The next step involves applying dummy coding to convert categorical variables into a format suitable for modelling. Finally, the process concludes with the removal of highly correlated predictors, which helps to reduce multicollinearity and improve model performance.

8.4.1 Performance on test data

Likewise, the resampled data, svmRadialCost model perform well on unseen data compared to other tree-based models.

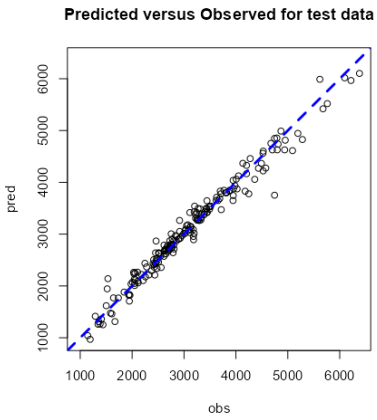

Figure 18: Predicted Vs Observed on Unseen Data - svmRadialCost model

Test Metric	Value
RMSE	174.077
MAE	119.307
R2	0.972

Table 8: Performance on Unseen Data - svmRadialCost model

Table 8 shows the test metrics for the svmRadialCost model, noting that the RMSE value is 174.077, which is lower than the RMSE value reported in the resampled performance. This indicates that the model performs better on the unseen test data than it did during the validation phase using resampling.

8.4.2 Residual Boxplots

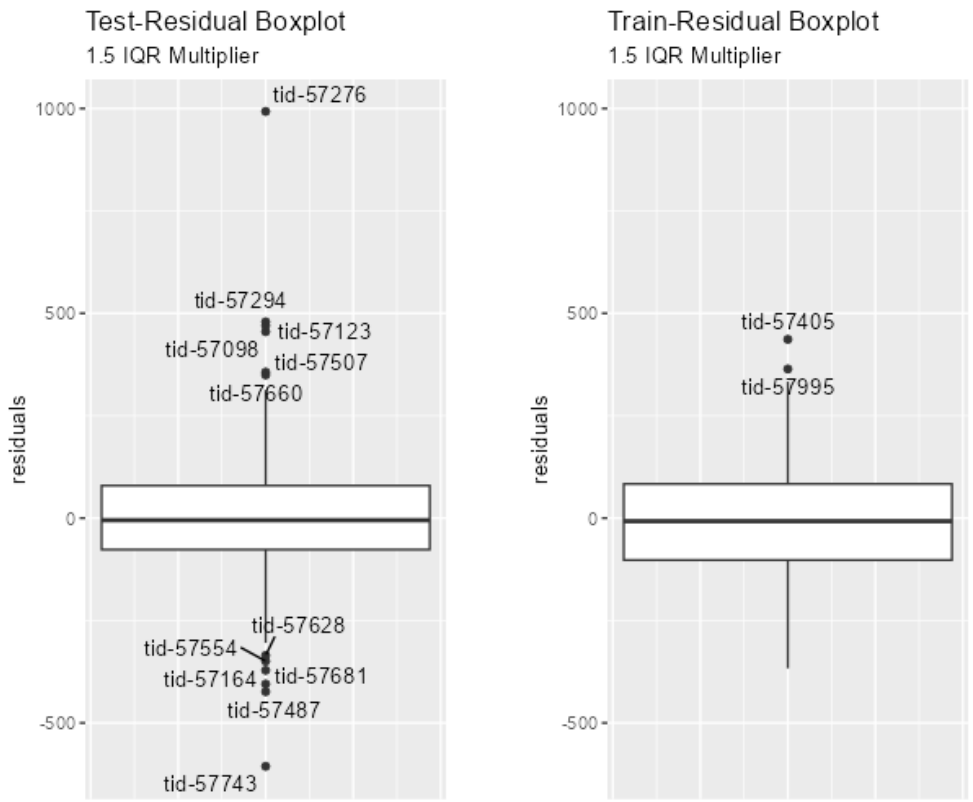

Figure 19: Residual Box Plots of svmRadialCost model

The residual box plots for the svmRadialCost method in figure 19 indicating that there are more residuals in test predictions than in train predictions.

8.5 Ensemble Models

Seven different ensemble models were initially tested, and the top five were chosen as candidate models for further optimization. The resampled performance of these selected models is presented in Figure 20 and Table 21.

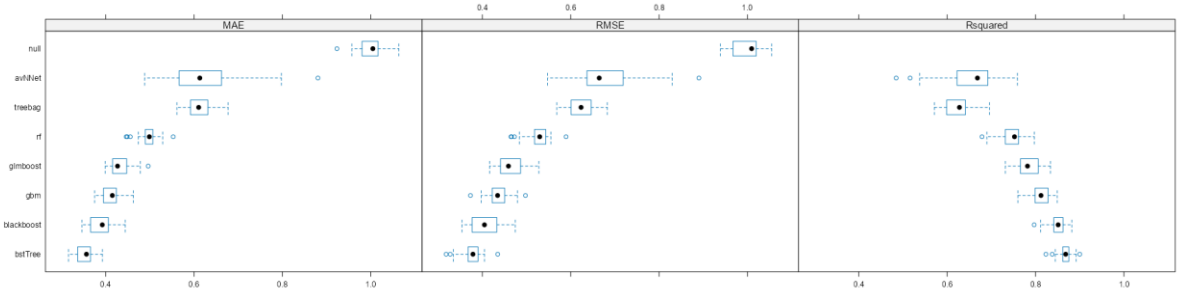

Figure 20: Resampled Performance of Ensemble models

Model	Processing steps	Hyper-parameters		Resampled Performance		
				RMSE	R2	MAE
bstTree	Impute kNN YeoJohnson Dow Dummy Corr	Maxdepth	3	389.52	0.87	282.00
		Nu	0.10			
		mstop	250			
blackboost	Impute kNN YeoJohnson Dow month Corr	Maxdepth	3	424.55	0.85	309.82
		mstop	200			
gbm	Impute_median Centre Scale Dummy Nzv corr	Shrinkage	0.10	451.69	0.81	328.77
		Interaction.depth	4			
		n.minobsinnode	10			
		n.trees	250			
glmboost	Impute Knn Dummy	Mstop	450	482.29	0.79	344.39
		Prune	no			
Random Forest	Impute Knn YeoJohnson	mtry	10	546.24	0.75	395.63

Table 9:Resampled Performance of Ensemble models

Out of these models, bst tree model performs better with the resampled data.

The preprocessing steps for the bst tree method begin with knn imputation, which is used to handle missing values. Then the YeoJohnson step is used to normalize the data. Following this, any date information is extracted and converted into a factor variable to facilitate categorical processing. The next step involves applying dummy coding to convert categorical variables into a format suitable for modelling. Finally, the process concludes with the removal of highly correlated predictors, which helps to reduce multicollinearity and improve model performance.

8.5.1 Performance on test data

Likewise, the resampled data, bst tree model perform well on unseen data compared to other Ensemble models.

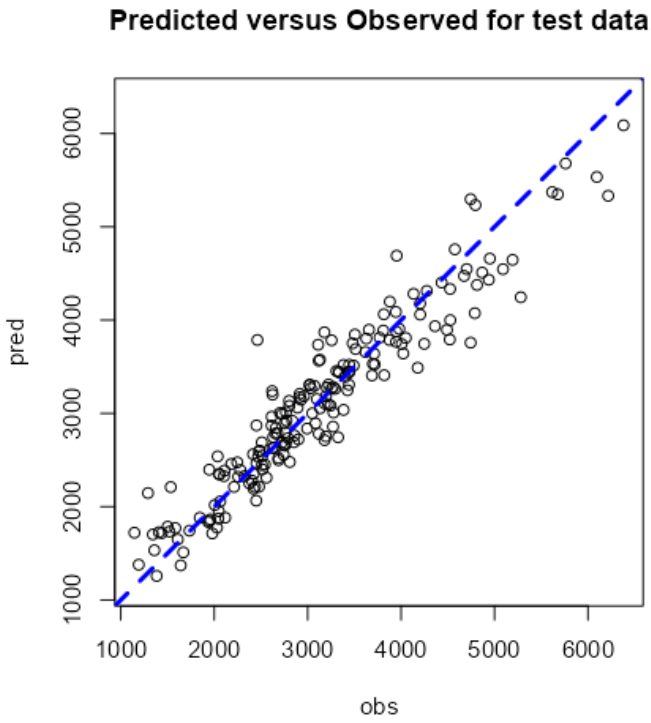

Figure 21: Predicted Vs Observed on Unseen Data - bstTree model

The performance on unseen data for bst tree model.

Test Metric	Value
RMSE	328.454
MAE	247.275
R2	0.903

Table 10: Performance on Unseen Data - bstTree model

Table 4 presents the test results for the bstTree model, indicating an RSME value of 328.5. This value is lower than the Resampled RSME value recorded for the bstTree method.

8.5.2 Residual Boxplots

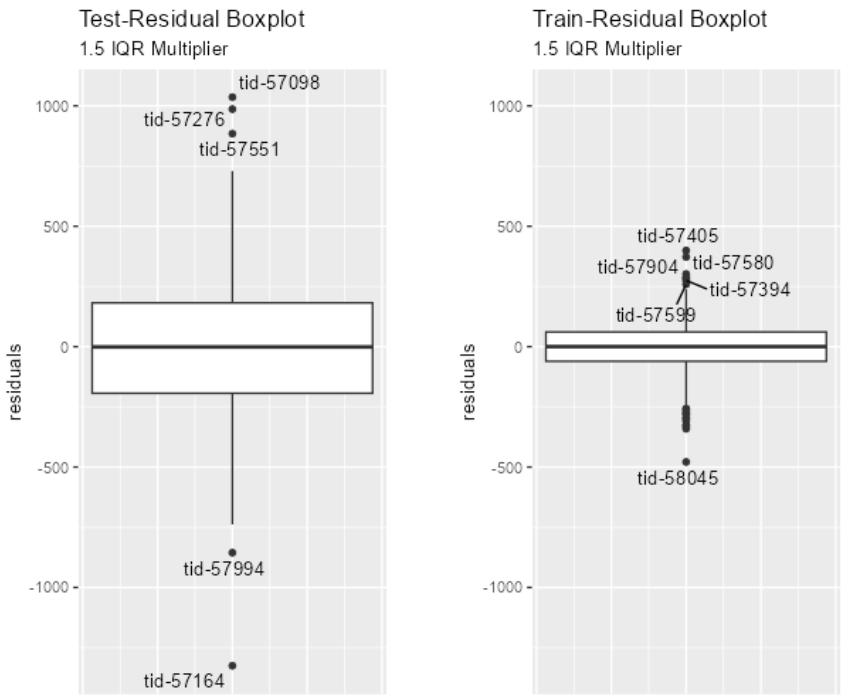

Figure 22: Residual Box Plots of bstTree model

According to the residual boxplots in figure 13, the bstTree model displays 5 test residuals and 6 train residuals when the interquartile range (IQR) value is set at 1.5.

8.6 Other Models

Out of all the trials, three methods did not fit into any of the previously mentioned categories. The resampled performances of these methods are displayed in Figure 24 and Table 11.

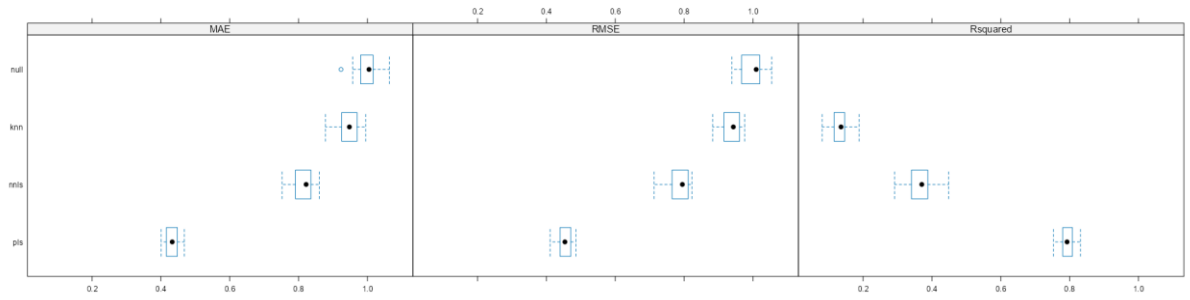

Figure 23: Resampled Performance of other models

Model	Processing steps	Hyper-parameters		Resampled Performance		
				RMSE	R2	MAE
pls	Na Omit YeoJohnson Dow dummy	ncomp	21	469.52	0.79	343.27
nnls	Na Omit Dow dummy			815.90	0.37	646.61
knn	Impute Knn Dow Dummy corr	k	45	970.57	0.13	750.43

Table 11: Resampled Performance of other models

According to the data on resample performance in Table 11, the PLS (Partial Least Squares) method exhibits the lowest RMSE (Root Mean Square Error) value.

8.6.1 Performance on test data

Likewise, the resampled data, pls model perform well on unseen data compared to other models.

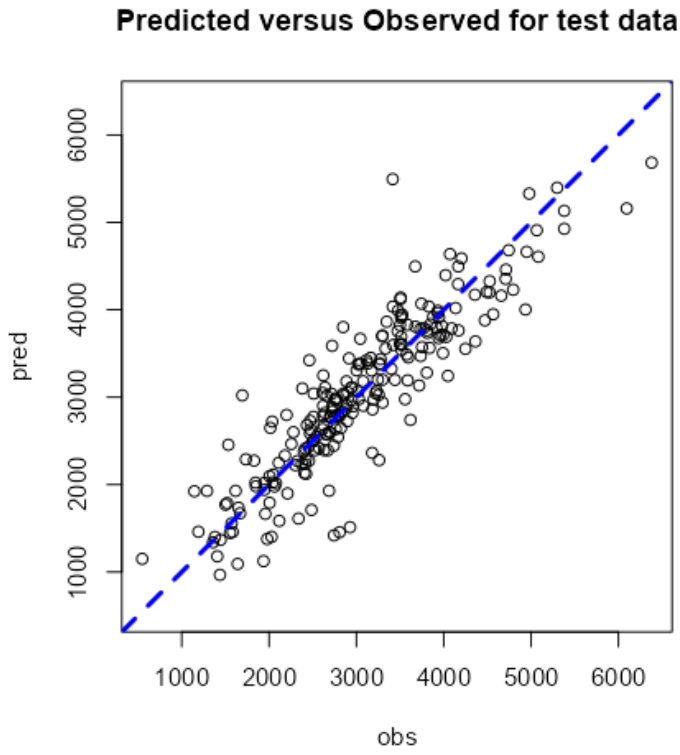

Figure 24: Predicted Vs Observed on Unseen Data - pls model

Test Metric	Value
RMSE	495.50
MAE	364.02
R2	0.78

Table 12: Performance on Unseen Data - pls model

Table 12 displays the test performance of the PLS (Partial Least Squares) model, indicating that the RMSE value is 495.50, which is higher compared to the RMSE value noted in the resampled performance.

8.6.2 Residual boxplots

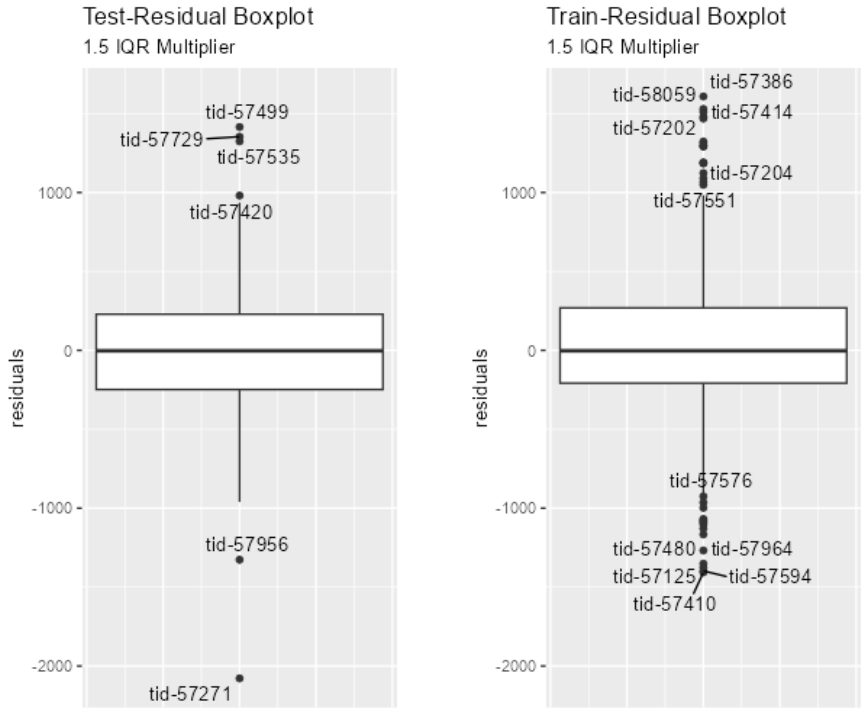

Figure 25: Residual Box Plots of pls model

According to the residual box plot shown in Figure 21, there are more residual outliers in the training predictions compared to the test predictions.

9 The Best Model

The following table show the best five models from all the methods which were tried above.

Model	Processing steps	Resampled Performance			Test Performance		
		RMSE	R2	MAE	RMSE	R2	MAE
monmlp	Impute bag Dow Dummy corr	118.50	0.99	89.53	94.978	0.992	67.221
svmRadialCost	Impute knn Dow Dummy Corr	228.94	0.95	156.06	174.077	0.972	119.307
svmRadial	Impute knn Dow Dummy	253.13	0.94	164.48	185.365	0.969	128.092
gaussprRadial	Impute knn Dow Dummy Corr	313.83	0.92	214.42	277.834	0.933	201.169
bst Tree	Impute kNN YeoJohnson Dow Dummy Corr	389.52	0.87	282	328.454	0.903	247.275

Table 13: Top five models

According to the table 13, monmpl neural network model has the lowest RMSE value in resample performance as well as in the test performance. The section 8.3 discusses more about the monmpl results, residuals and preprocessing steps.

There are several reasons why the monmpl neural network model performed best in terms of RMSE values in both test and resample trials. First off, it fits the data more precisely because it can represent complex nonlinear relationships in the data that simpler models can miss. The particulars of the dataset, especially if it contains high dimensional features or complicated patterns, may also make neural networks' flexible modelling abilities a better fit (Lang, 2005).

9.1 monmpl neural network

Monmlp is a multi-layer perceptron (MLP) neural network with one or two hidden layers. It offers an optional feature that allows you to impose a monotone constraint on model outputs, guaranteeing that they rise monotonically in respect to specified variables. This feature produces a monotone MLP (MONMLP) regression model based on the methods published by Zhang and Zhang in 1999 (Cannon, 2017).

10 Best Performing Transparent Model

10.1 Transparent Model

A "transparent model" in machine learning and artificial intelligence is defined by its simplicity of understanding and interpretability, allowing users to see how inputs are translated into outputs and comprehend the decision-making process. Such models, which include decision trees, linear regression, and logistic regression, have simpler processes that allow for easier tracing of their activities without extensive computations that hide their rationale (Zhou Y, 2020).

10.2 The best transparent model

While the monmpl model stands out based on it's resample performance, it is not considered a transparent model. Therefore, the following table presents the best five transparent models from all the methods previously tested. This table aims to highlight models that not only perform well but also provide clear and understandable decision-making processes, allowing for easier interpretation and validation of results.

Model	Processing steps	Resampled Performance			Test Performance		
		RMSE	R2	MAE	RMSE	R2	MAE
rlm	Na omit Dow Dummy corr	462.85	0.80	330.11	493.477	0.772	364.179
glmnet	Na omit YeoJohnson Dummy	472.59	0.79	340.31	515.201	0.758	380.374
leapSeq	Na omit YeoJohnson month Dummy	479.98	0.78	347.49	528.778	0.748	389.191
M5	Impute_knn Dow month	478.31	0.79	344.85	505.156	0.769	378.269
M5Rules	Impute_knn Dow month	480.24	0.79	346.23	505.156	0.769	378.269

Table 14: Top five transparent models

According to the table 14, the rlm(robust linear model) model has the lowest resampled and test RSMEs, making it the best transparent model among the lot for this dataset. The section 8.1 discusses more about the monmpl results and residuals.

11 Conclusion

In this assignment, over 30 different regression methods from the caret package were examined to determine the best model. These 32 approaches were grouped into five primary categories, with any methods that did not fall into one of these groups being classed as "other." This technique helps to organise the methodologies and speed the process of assessing their performance.

Most of the base methods for the ensemble methods were initially part of one of the other specified categories. However, due to their ensemble characteristics, these models were categorized separately to distinguish them from transparent methods. Ensemble methods, which are not inherently transparent, are grouped into their own category if the description includes the word "Ensemble," regardless of their original classification.

During the evaluation of the best models in this experiment, the monmlp model stood out as the most effective, achieving the lowest RMSE values in both the resampled and test performances. As a result, this model was considered the most effective. Additionally, three other kernel methods ranked among the top five models, indicating their strong performance and reliability in this set of experiments. These approaches' effectiveness is partly due to their ability to handle nonlinear relationships between predictors.

When evaluating the transparent models, the rlm (robust linear model) stood out by achieving the lowest RMSE values in both resampled and test performances. This indicates that when transparency is a priority, valuing the ability to understand and interpret the model's decision-making process, the robust linear model is the optimal choice for achieving the best results.

The whole process exhibits the "no-free lunch" theory, which states that no single method is universally optimal for all scenarios. It highlights the importance of selecting the optimum model based on the dataset's unique characteristics and desired outcomes. This approach emphasises the significance of customising model selection to the specific needs and goals of each project.

12 References

Agarrwal, A. (2017, April 10). *R Shiny Tutorial*. Retrieved from Data Science Tutorials: https://www.youtube.com/watch?v=KdvlxJaWWVQ&t=515s

Boehmke, B., & Greenwell, B. (2020, February 01). *Hands-On Machine Learning with R.* Retrieved from Feature & Target Engineering: https://bradleyboehmke.github.io/HOML/

Cannon, A. J. (2017). *Multi-Layer Perceptron Neural Network with Optional Monotonicity.*

Lang, B. (2005). *Monotonic Multi-layer Perceptron Networks as Universal Approximators.* Duch, W., Kacprzyk: Springer, Berlin, Heidelberg.

OpenAI. (2024, March 01). *ChatGTP*. Retrieved from ChatGTP: https://chat.openai.com/

QuillBot. (2024, March 01). *QuillBot*. Retrieved from QuillBot: https://quillbot.com/

Shiny. (2024, March 1). *Shiny for R*. Retrieved from Shiny : https://shiny.posit.co/r/deploy

Stack Exchange Inc. (2024). *stackoverflow*. Retrieved from stackoverflow: https://stackoverflow.com/

Zhou Y, K. M. (2020). *On Transparency of Machine Learning Models: A Position Paper.* Harward University.