Forecasting with Machine Learning Models

Nicholas Bradley

# Forecasting with Machine Learning models

This analysis focuses on using machine learning models for time series forecasting using the h2o package and covers the following topics:

* Introduction to the h2o package and its functionality
* Feature engineering of time series data
* Forecasting with the Random Forest model
* Forecasting with the gradient boosting model
* Forecasting with the automate model

## When and why should machine learning be used:

Recently, machine learning (ML) models have become popular and accessible to significant improvements in computing power, which led to a variety of new methods becoming available. It is important to contextualise ML models in time series forecasting:

* *Cost:* The use of ML models is generally more expensive than typical regression models both in computing time and power.
* *Accuracy:* ML model performance is highly dependent on the quality(that is, strong casuality relationship with the dependent variable) of the predictors. Its likely that ML models will overperform, with respect to traditional methods when quality predictors are available.
* *Tuning:* Processing typical ML models is more complex than with traditional regression models, as those models have more tuning parameters and therefore, require some expertise.
* *Black-Box:* Most ML models are considered black boxes, as it is hard to interrupt their output
* *Uncertainty:* Generally, there is no straightforward method to quantify the forecasting uncertainty with confidence intervals like the traditional time series model does.

The advantage of ML models is their predictive power, which in many cases, is worth the time and effort involved in the process. In the context of time series forecasting, it is beneficial to forecast with ML models in the following cases:

* *Structural patterns:* Exits in the series, as those can produce new, predictive features = *Multiple seasonality:* As a special case for structural patterns since, typically, the traditional time series model struggles to capture those patterns when they exist.

## Why H2o?

H2o is an open source java based library for ML applications including supervised and unsupervised ML models in both R and python programming. H2o is based on distributed processing, so can either be used in memory or with some external computing power. The algorithms in H2o provide several methods that we can train and tune ML models, such as cross validation method and the built in grid search function.

## Forecasting monthly vehicle sales in the US

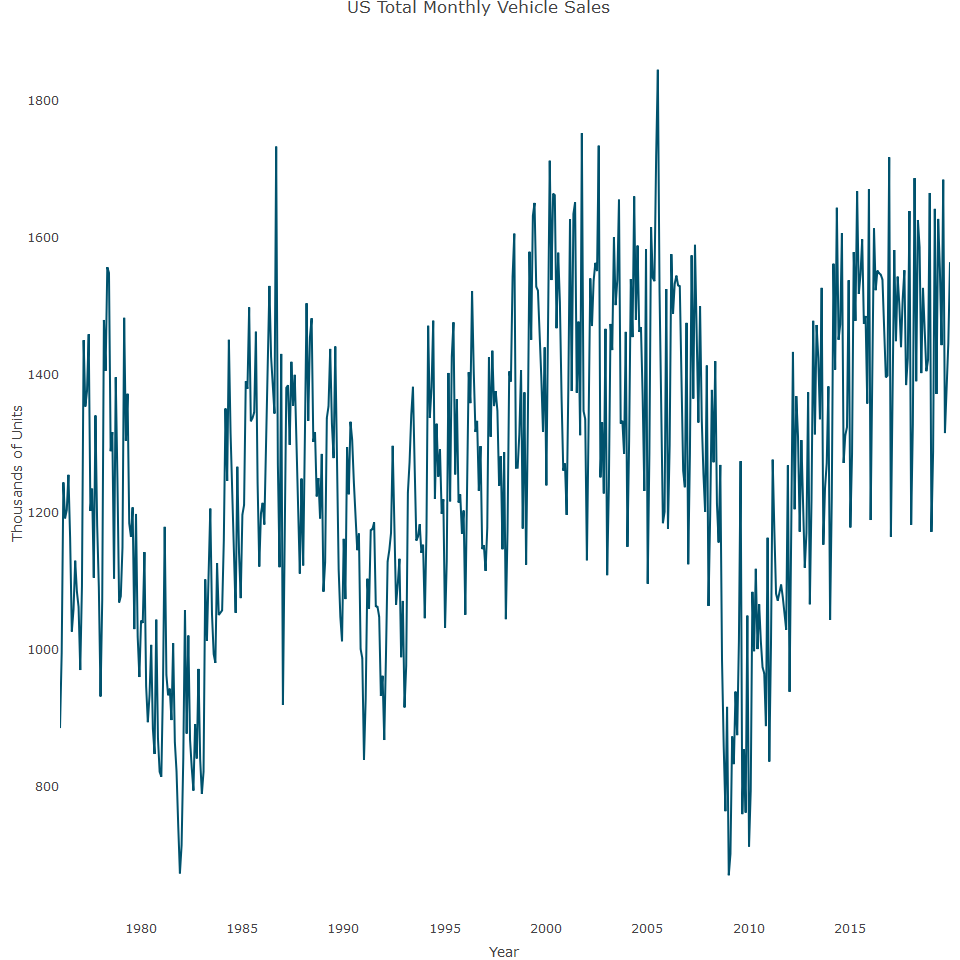
We will focus on forecasting the total monthly sales in the US in the next 12 months using ML methods. Before starting to prepare the series and create new features, we can carry out a quick exploratory analysis of the series to identify the main series characteristics, including:

* View the time series structure (frequency, start and end of the series, etc)
* Explore the seasonal components (seasonal, cycle, trend and random components)
* Seasonality analysis
* Correlation analysis

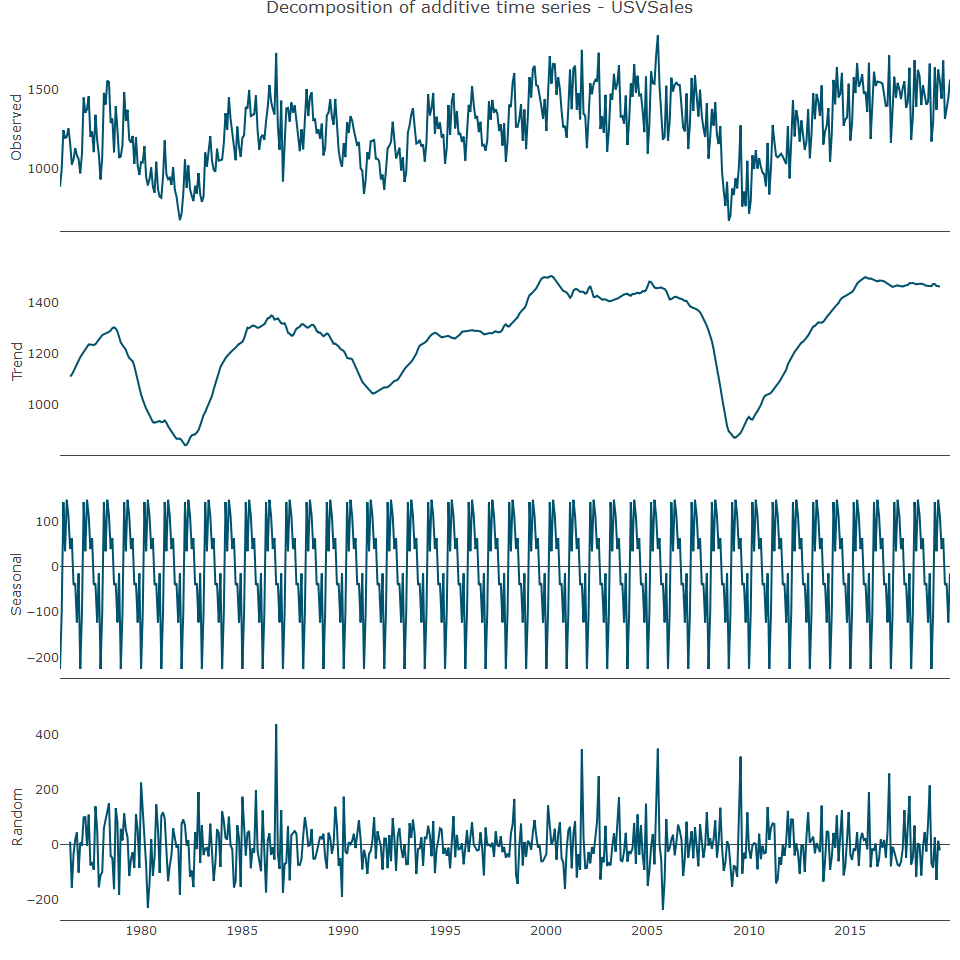
We can start with reviewing the structure of the series

## The USVSales series is a ts object with 1 variable and 528 observations  
## Frequency: 12   
## Start time: 1976 1   
## End time: 2019 12

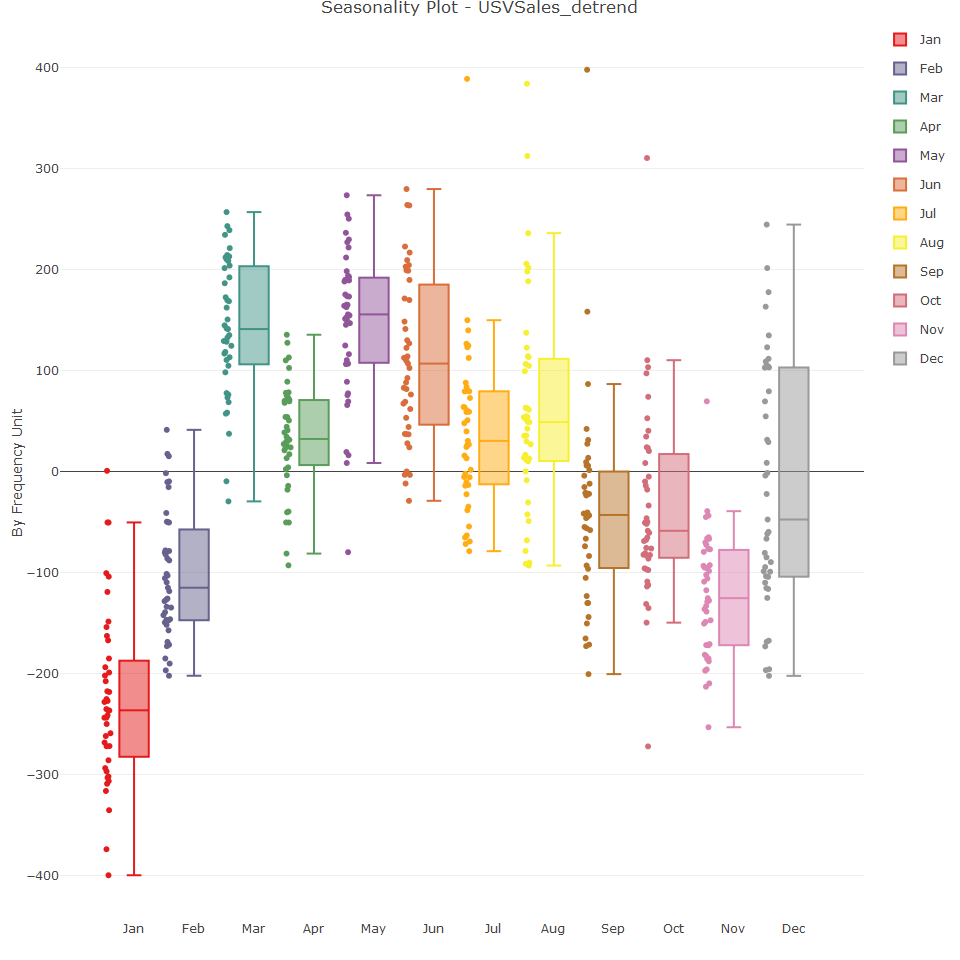
The USVSales series is a monthly time series object which represents the total vehicle sales in the US between 1976 and 2018 in thousands of units. We can plot it:



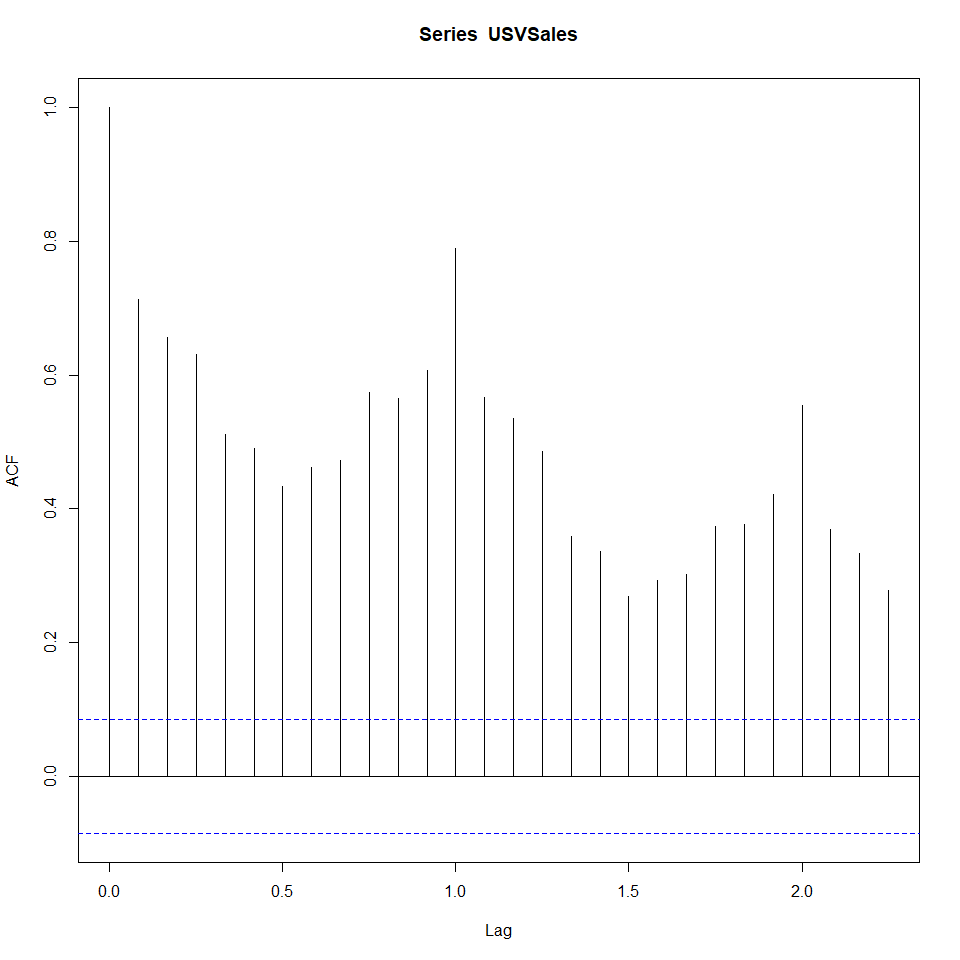
The plot indicates the series has cycle patterns, which is common for a macro economy indicator, which in this case, it is a macro indicator of the US economy. We can get a better understanding of the series components by decomposing it and plotting it:



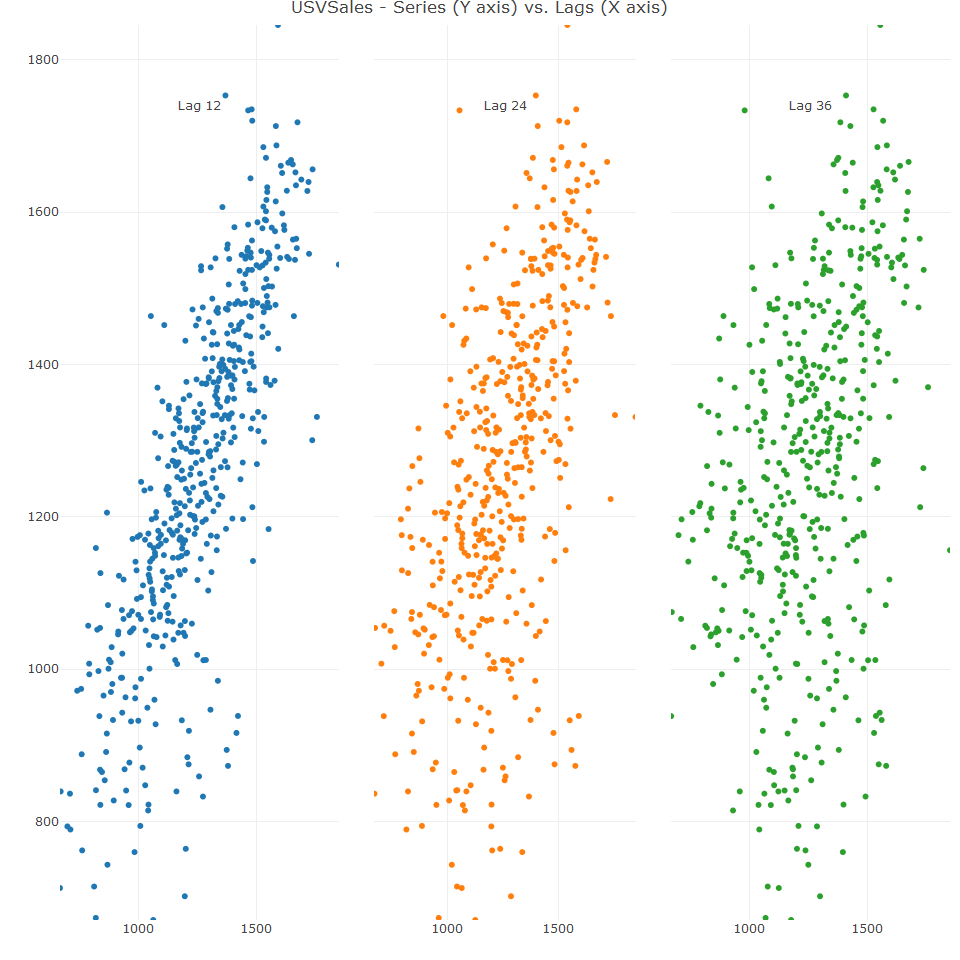
Beside the cycle-trend component, we can observe that the plot has a strong seasonal pattern, which we can explore next. To get a closer look at the seasonal component of the series, we can subtract from the series, decompose the trend and plot the box plot of the seasonal component of the detrend series:



We can see in the plot above that typically, the peak of the year occurred during the months of March, May and June, and that the sales decay from the summer months and peak again December during the holiday season. However, January is the lowest month of the year for sales. We can now complete a correlation analysis and as seen in the plot below, the series has a high correlation with its first seasonal lag.



We can zoom in on the relationship of the series with the last three seasonal lags using the ts\_lags function



As indicated in the above plot, the relationship of the series with the first and second lag has a strong linear relationship.

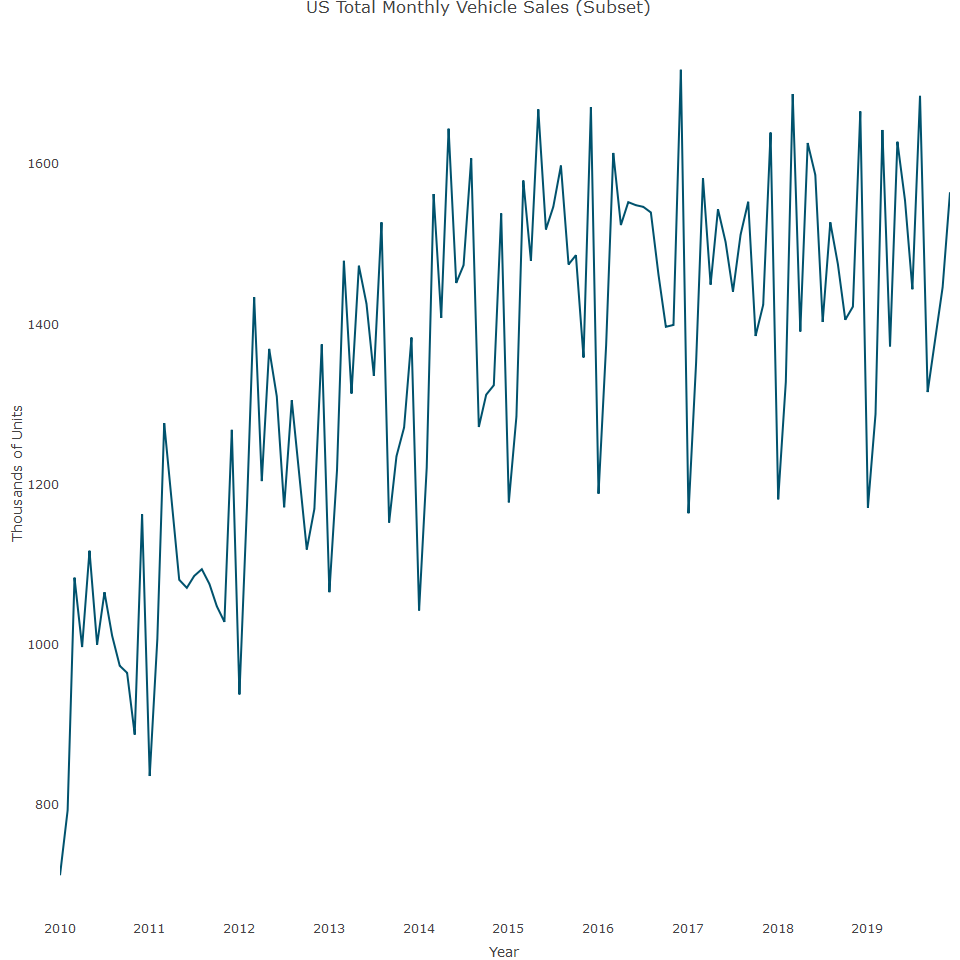
## Exploratory analysis - key findings:

The exploratory analysis has the following findings:

* The USVSales series is a monthly series with a clear monthly seasonality
* The series trend has a cyclic shape, and so, the series has a cycle component embedded in the trend
* The series’ most recent cycle starts right after the end of the 2008 economic crisis, between 2009 and 2010
* It seems that the current cycle reached its peak as the trend started to flatten out - The series has a strong correlation with its first seasonal lag

As we intend to have short-term forecast (of 12 months) there is no point in using the full series, as it may enter in some noise into the model due to the change of the trend direction every couple of years. If we wanted to create a long-term forecast, then it may be a good idea to use all or most of the series. Therefore, we will use the model training observations from 2010 and onward.

Lets plot the time series:



## Feature Engineering

Feature engineering plays a pivotal role when modelling with ML algorithms. The next step, based on preceding observations is to create new features that can be used as informative input for the model. In the context of time series forecasting, here are some possible new features that can be created from the series itself:

* *The series trend:* This uses a numeric index. In addition, as the series trend isn’t linear, we will use a second polynomial of the index to capture the overall curvature of the series trend.
* *Seasonal component:* This creates a categorical variable for the month of the year to capture the series’ seasonality.
* *Series correlation:* This utilises the strong correlation of the series with its seasonal lag and uses the seasonal lag (lag 12) as an input to the model.

## 'data.frame': 108 obs. of 6 variables:  
## $ date : Date, format: "2011-01-01" "2011-02-01" ...  
## $ y : num 836 1007 1277 1174 1081 ...  
## $ month : Factor w/ 12 levels "Jan","Feb","Mar",..: 1 2 3 4 5 6 7 8 9 10 ...  
## $ lag12 : num 712 793 1084 997 1118 ...  
## $ trend : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ trend\_sqr: num 1 4 9 16 25 36 49 64 81 100 ...

## Training, testing and model evaluation

Since our forecast horizon is 12 months, we will leave the last 12 months of the series as testing partitions and use the rest of the series as a training partition:

We can evaluate model performance using the MAPE score on the testing partition. A main characteristic of ML models is the tendency to overfit on a training set. Therefore, you should expect that the ratio between the error score on the training and test data will be relatively larger than the traditional times series models, such as Holt-Winter and time series linear regression. In addition to the training and testing partitions, we need to create inputs for the forecast itself. We can create a data frame with the dates of the following 12 months and build the rest of the features.

## Model benchmark

The performance of a forecasting model should be measured by the error rate, mainly on the testing partition, but also on the training data and the model performance should be evaluated with respect to some baseline model - for example seasonal naive model. Since we are using a family of ML regression models, it makes sense to use a regression model as the benchmark, such as time series linear regression model. Let’s now train the linear regression model and evaluate performance on testing data

##   
## Call:  
## lm(formula = y ~ month + lag12 + trend + trend\_sqr, data = train\_df)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -146.625 -38.997 0.111 39.196 112.577   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 542.93505 72.59490 7.479 7.91e-11 \*\*\*  
## monthFeb 112.73160 34.16141 3.300 0.001439 \*\*   
## monthMar 299.20932 54.24042 5.516 4.03e-07 \*\*\*  
## monthApr 182.52406 42.53129 4.292 4.88e-05 \*\*\*  
## monthMay 268.75603 51.28464 5.240 1.24e-06 \*\*\*  
## monthJun 224.66897 44.26374 5.076 2.41e-06 \*\*\*  
## monthJul 177.88564 42.21898 4.213 6.49e-05 \*\*\*  
## monthAug 241.63260 47.00693 5.140 1.86e-06 \*\*\*  
## monthSep 152.99058 37.04199 4.130 8.76e-05 \*\*\*  
## monthOct 125.16484 35.04896 3.571 0.000601 \*\*\*  
## monthNov 127.97288 34.18772 3.743 0.000338 \*\*\*  
## monthDec 278.67994 51.09552 5.454 5.21e-07 \*\*\*  
## lag12 0.33906 0.10738 3.158 0.002236 \*\*   
## trend 7.73667 1.72415 4.487 2.36e-05 \*\*\*  
## trend\_sqr -0.05587 0.01221 -4.576 1.69e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 59.6 on 81 degrees of freedom  
## Multiple R-squared: 0.9198, Adjusted R-squared: 0.9059   
## F-statistic: 66.36 on 14 and 81 DF, p-value: < 2.2e-16

## Interpretation of the output:

* *Multiple R-squared:* Indicates that approximately 91.98% of the variability in the response variable y is explained by the predictors.
* *Adjusted R-squared:* Adjusts the R-squared value for the number of predictors in the model, providing a more accurate measure for models with multiple predictors. Here it is 90.59%.
* Given the F-statistic of 66.36 and the p-value < 2.2e-16, we can confidently reject the null hypothesis. This means that, collectively, the predictor variables (month, lag12, trend, trend\_sqr) have a statistically significant effect on the response variable y.

## Specific Implications:

* *Model Significance:* The very low p-value (< 2.2e-16) indicates that the model as a whole is statistically significant. This means that the predictors included in the model explain a significant portion of the variability in the response variable y.
* *Predictor Contributions:* The significant F-statistic suggests that at least one of the predictors is contributing to the model’s explanatory power. Looking at the individual coefficients and their p-values, most predictors (month indicators, lag12, trend, trend\_sqr) are significant at the 0.01 level or better.
* *Practical Use:* Given the model’s significance, it can be used to make reliable predictions about y based on the included predictors. The high R-squared value (90.59%) further supports that the model explains a substantial amount of the variability in the data, making it a useful tool for understanding the relationship between the predictors and the response variable.

Next, we can predict corresponding values of the series on the testing partition:

## [1] 0.03594578

The MAPE score is 3.5% for the linear regression model, which we can use as benchmark.

## Starting a H2o cluster:

This package is based on use of distributed and parallel computing in order to speed up compute time and be able to scale up for big data, which can be done on a local computer’s RAM or on cloud based solutions such as AWS, Google Cloud etc. When the package is loaded, we set the in-memory cluster with the h2o.init function.

## Connection successful!  
##   
## R is connected to the H2O cluster:   
## H2O cluster uptime: 52 minutes 59 seconds   
## H2O cluster timezone: Europe/London   
## H2O data parsing timezone: UTC   
## H2O cluster version: 3.44.0.3   
## H2O cluster version age: 5 months and 25 days   
## H2O cluster name: H2O\_started\_from\_R\_Nicholas\_dfv347   
## H2O cluster total nodes: 1   
## H2O cluster total memory: 15.56 GB   
## H2O cluster total cores: 16   
## H2O cluster allowed cores: 16   
## H2O cluster healthy: TRUE   
## H2O Connection ip: localhost   
## H2O Connection port: 54321   
## H2O Connection proxy: NA   
## H2O Internal Security: FALSE   
## R Version: R version 4.3.1 (2023-06-16 ucrt)

## Warning in h2o.clusterInfo():   
## Your H2O cluster version is (5 months and 25 days) old. There may be a newer version available.  
## Please download and install the latest version from: https://h2o-release.s3.amazonaws.com/h2o/latest\_stable.html

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

We need to label the names of the dependent and independent variables:

## Forecasting with the Random Forest Model

We are now ready to build our first forecasting model with the Random Forest model.Its one of the more popular models and can be used for classification and regression. As its name implies, there are two parts to it:

* *Random:* The input for each tree model is based on a random sample, along with the replacement of both the columns and rows of the input data. This is known as bagging.
* *Forest:* The collection of tree based models is known as the forest

Explanation:

* *rf\_md:* This is the variable that will store the trained Random Forest model.
* *h2o.randomForest:* This function from the H2O library is used to create and train a Random Forest model.

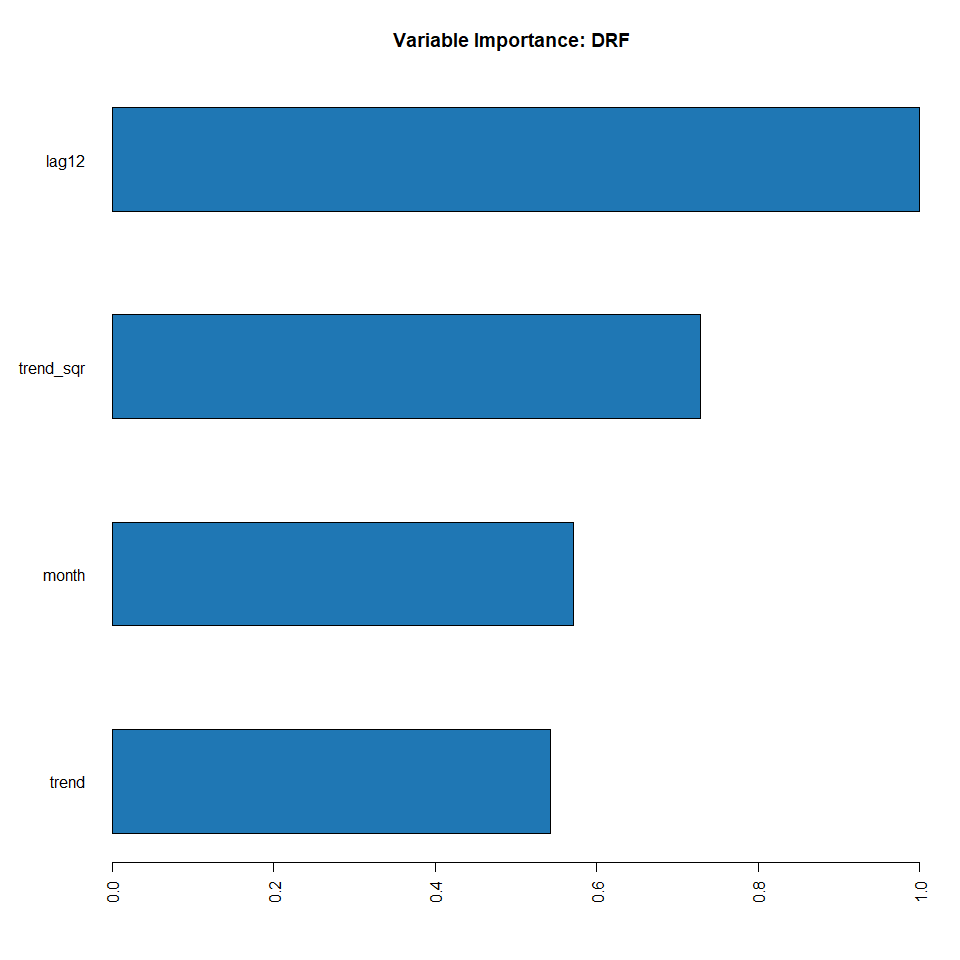
Parameters - *training\_frame = train\_h:* Specifies the training dataset for the model. train\_h should be an H2OFrame containing the data you want to use for training.

* *nfolds = 5:* Sets the number of folds for cross-validation. This means the dataset will be divided into 5 parts, and the model will be trained 5 times, each time using a different part as the validation set and the remaining parts as the training set. This helps in getting an estimate of the model’s performance.
* *x = x:* Specifies the feature columns used for training the model. x should be a vector of column names or indices from the training\_frame.
* *y = y:* Specifies the target column that the model is trying to predict. y should be the name or index of the response variable in the training\_frame.
* *ntrees = 500:* Sets the number of trees to build in the Random Forest. More trees can increase accuracy but also increase computation time.
* *stopping\_rounds = 10:* Enables early stopping based on a specified metric. If the metric doesn’t improve for 10 consecutive scoring rounds, training will stop.
* *stopping\_metric = “RMSE”:* Specifies the metric used to evaluate model performance during training. RMSE (Root Mean Squared Error) is used here, which is commonly used for regression tasks to measure the average magnitude of the errors between predicted and actual values.
* *score\_each\_iteration = TRUE:* Instructs the model to score and evaluate the model on the validation data at each iteration. This is useful for monitoring the model’s performance during training.
* *stopping\_tolerance = 0.0001:* Sets the tolerance for the stopping metric. Training will stop if the improvement in the stopping metric is less than this value over the specified stopping\_rounds.
* *seed = 1234:* Sets a random seed to ensure reproducibility of the results. Using the same seed will produce the same sequence of random numbers, which helps in getting consistent results across different runs.

## Summary

This code trains a Random Forest model using the H2O library in R. It uses 500 trees and 5-fold cross-validation to evaluate the model’s performance. Early stopping is enabled to prevent overfitting, using RMSE as the stopping metric. The model is trained using specified features (x) and target variable (y), and the results are reproducible due to the fixed random seed.

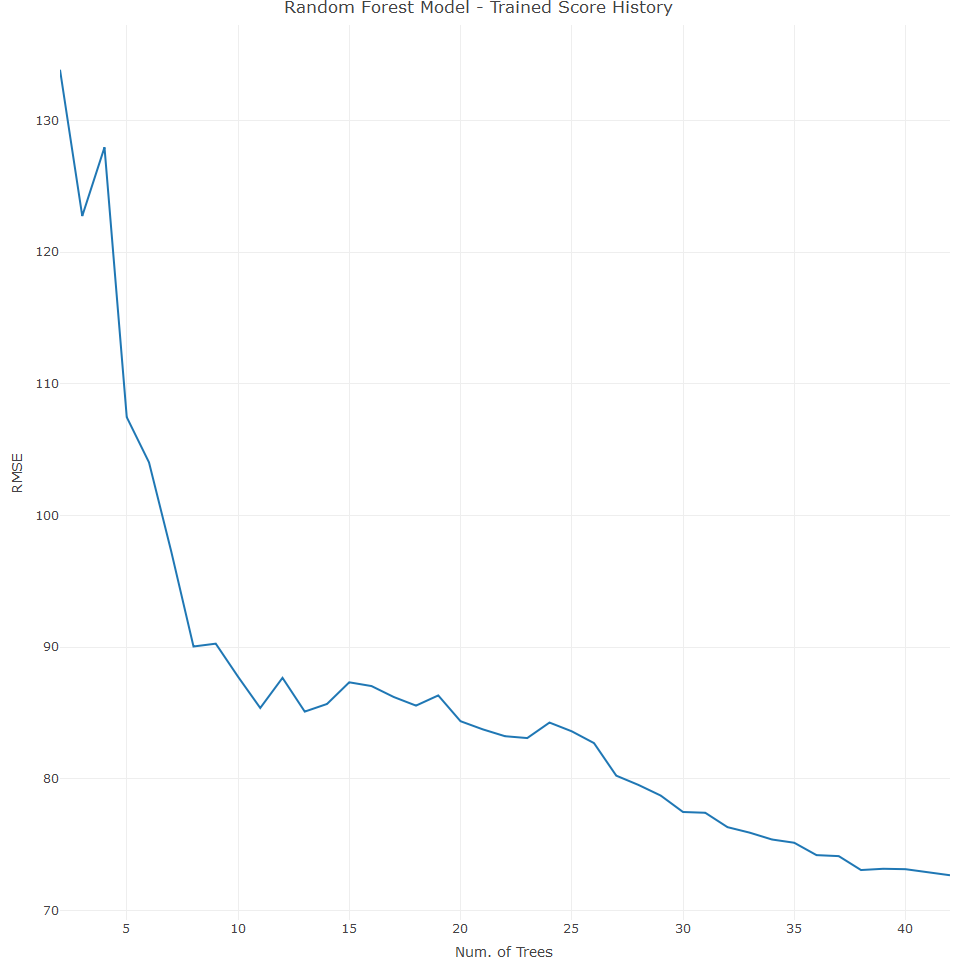
Now let us see which variables are most important:



The most important variable is LAG12, which was indicated in the correlation analysis as it showed a strong relationship between the series and the seasonal lag. Let’s review the summary:

## Model Summary:   
## number\_of\_trees number\_of\_internal\_trees model\_size\_in\_bytes min\_depth  
## 1 41 41 31590 8  
## max\_depth mean\_depth min\_leaves max\_leaves mean\_leaves  
## 1 12 10.04878 45 66 56.70732

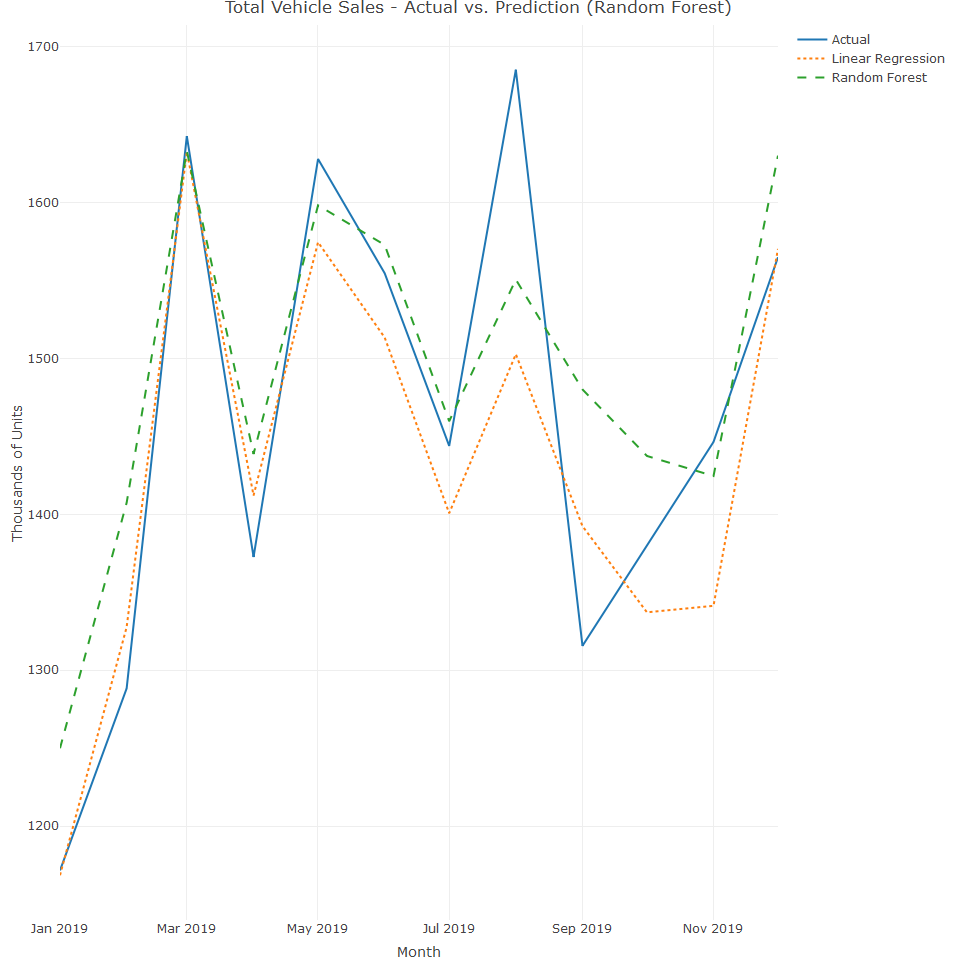
We can see that the model utilized 35 out of the 500 trees that were set by the ntrees argument.The following plot indicates the learning process of the model as a function of the number of trees.



We can measure the model’s performance on the testing partition:

## [1] 0.04647164

The MAPE score is 3.5% for the linear regression model, which we use as benchmark. The random forest model is 4.6%, so it has a higher error, so doesn’t perform as well as the benchmark model.



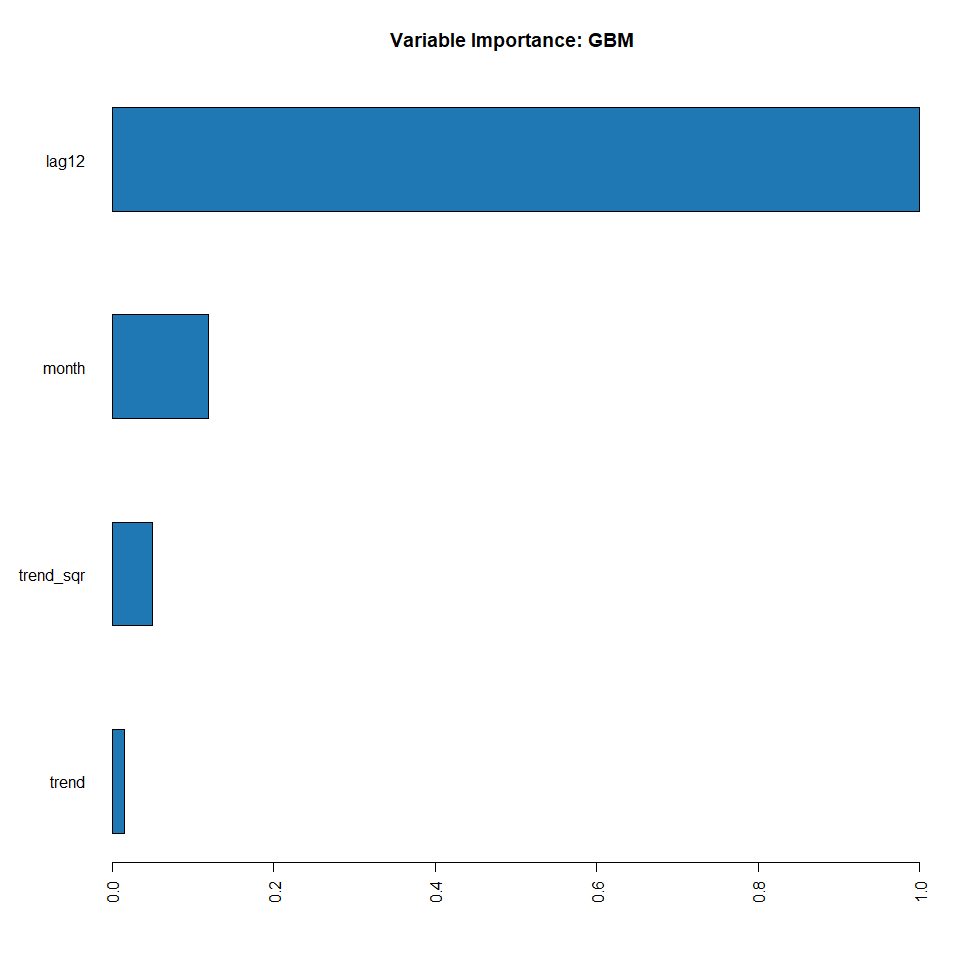
## Forecasting with the gbm model

This algorithym is another ensemble and tree based model.It uses the boosting approach in order to train different subsets of the data and repeats the training of subsets that the model had with a high error rate. This allows the model to learn from past mistakes and improve the predictive power of the model. The main arguments of a GBM model are:

* formula specifies that response is the dependent variable and all other variables in data are predictors.
* distribution specifies that the response variable follows a Gaussian distribution (appropriate for regression problems).
* data specifies the dataset to be used for training.
* n.trees specifies the number of boosting iterations.
* interaction.depth sets the maximum depth of each tree.
* n.minobsinnode sets the minimum number of observations in the terminal nodes.
* shrinkage sets the learning rate.
* cv.folds specifies the number of cross-validation folds.
* train.fraction specifies the proportion of the data will be used for training.

This setup will train a GBM model and determine the optimal number of trees based on cross-validation results. Adjust these parameters according to the specifics of your dataset and problem to optimize the performance of your model.

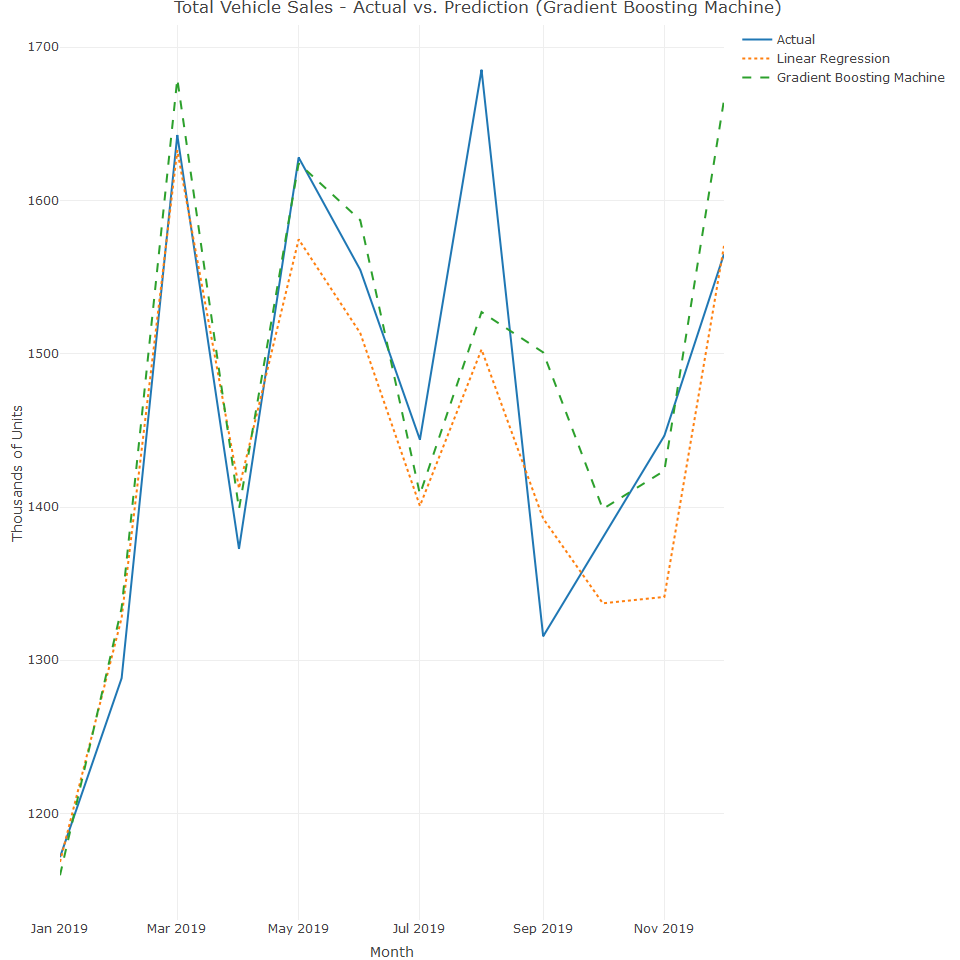
Similar to the Random Forest, we can see the importance of each variable:



Again, we can see that LAG12 is the most important variable. Let’s test the model’s performance:

## [1] 0.03857898

The MAPE score is 3.8%, which is an improvement on the random forest models, the basic random forest model 4.6% and the grid search random forest (3.9%). However,the gbm model performs worse than the linear regression with its MAPE score of 3.5%. Let’s visualise the GBM model against the other models:



## Forecasting with the AutoML model:

We can use the h2o.automl function which provides an automated approach to training, testing and tuning multiple algorithms before selecting the best performing model.

We can see the leading models with the below code:

## model\_id rmse mse  
## 1 StackedEnsemble\_BestOfFamily\_5\_AutoML\_3\_20240615\_140440 58.33091 3402.495  
## 2 StackedEnsemble\_BestOfFamily\_3\_AutoML\_3\_20240615\_140440 59.72582 3567.174  
## 3 StackedEnsemble\_AllModels\_6\_AutoML\_3\_20240615\_140440 59.98812 3598.574  
## 4 StackedEnsemble\_AllModels\_3\_AutoML\_3\_20240615\_140440 61.88866 3830.206  
## 5 GBM\_grid\_1\_AutoML\_3\_20240615\_140440\_model\_126 62.44813 3899.769  
## 6 StackedEnsemble\_AllModels\_4\_AutoML\_3\_20240615\_140440 62.86863 3952.464  
## mae rmsle mean\_residual\_deviance  
## 1 47.65586 0.04495635 3402.495  
## 2 48.06552 0.04553704 3567.174  
## 3 48.81093 0.04639338 3598.574  
## 4 48.73172 0.04802818 3830.206  
## 5 50.49076 0.04852091 3899.769  
## 6 49.56460 0.04925605 3952.464  
##   
## [242 rows x 6 columns]

Let’s now test the model’s performance on the test set:

| ## [1] 0.04421454

The leading model in the h2o.automl achieved a MAPE of 3.4%. The gbm MAPE score is 3.8%, which is an improvement on the random forest model, the basic random forest model 4.6% . However, h2o.automl model performs better than the linear regression with its MAPE score of 3.5%.

## Selecting the final model

The leading model in the h2o.automl achieved a MAPE of 3.4%. The gbm MAPE score is 3.8%, which is an improvement on the random forest models, the basic random forest model 4.6%. However, h2o.automl model performs better than the linear regression with its MAPE score of 3.5%. Let’s choose the h2o.automl model, the gbm mode. We can ignore the linear regression model as it can’ be predicted using h2o.predict function.