# Customer Acquisition Case Study

Cynthia Farias, Maria Morinigo, Robert Perez, Rubina Saya

4/23/2021

## Executive Summary

In this case study, the aim is to fit several predictive models to find the best model that accurately predicts the acquisition of new customers. Our analysis resulted in the `random forest` model providing the highest accuracy rate of 82.29%. This accuracy was compared to logistic and decision tree models, which resulted in an accuracy of 80% and 71.33%, respectively. Even though the `logistic model` 's accuracy is comparable, the `random forest` was selected as the final model due to its ability to deal with multicollinearity issues, non-linear relationships, missing values, outliers, and unbalanced data.

Further, it was noted that the most significant variable identified using the three models consistently included `industry` , `revenue` , and `employees` , with random forest also identifying `acq_exp` as having significance.

On customer observations predicted as `acquired` using the `random forest` model, another tuned `random forest` was utilized to identify the most significant variables for customer duration. The model resulted in the `ret_exp_sq` , `ret_exp` , `freq_sq` , and `freq` variables being the most significant in determining the duration of the customer with the company.

Overall, the `random forest` proved to be a reliable model in predicting customer acquisition. The resulting information from this case study related to customer acquisition and duration will provide companies with valuable insights to establish informed targeted marketing strategies to gain business, reduce cost, and increase profitability.

## Problem

For a company to be successful, it must understand which variables impact customer acquisition and retention. The purpose of this case study is to identify the best model to predict customer acquisition and then to apply the selected model to classify the most significant explanatory variables impacting both customer acquisition and duration. This will allow the company to take appropriate actions to ensure that customers are acquired and then invest in resources to increase the customers' tenure with the company. The company may use the knowledge gained to deploy targeted marketing strategies, including discounts and other incentives, to gain and retain customers.

The goal of this case study is to answer the following questions:

- Which predictive model between Random Forest, Logistic, and Decision Tree can predict customer acquisition Retention data set to predict which customers will be acquired?
- Which variables are significant in determining new customer acquisition and duration of the customers with the company?
- If a customer is acquired, for how long is that customer retained?

The remaining sections of this report will look at the existing literature related to models used to predict customer retention and churn, discuss the data exploration and preprocessing, methodologies applied to develop the best model, and present the results and recommendations.

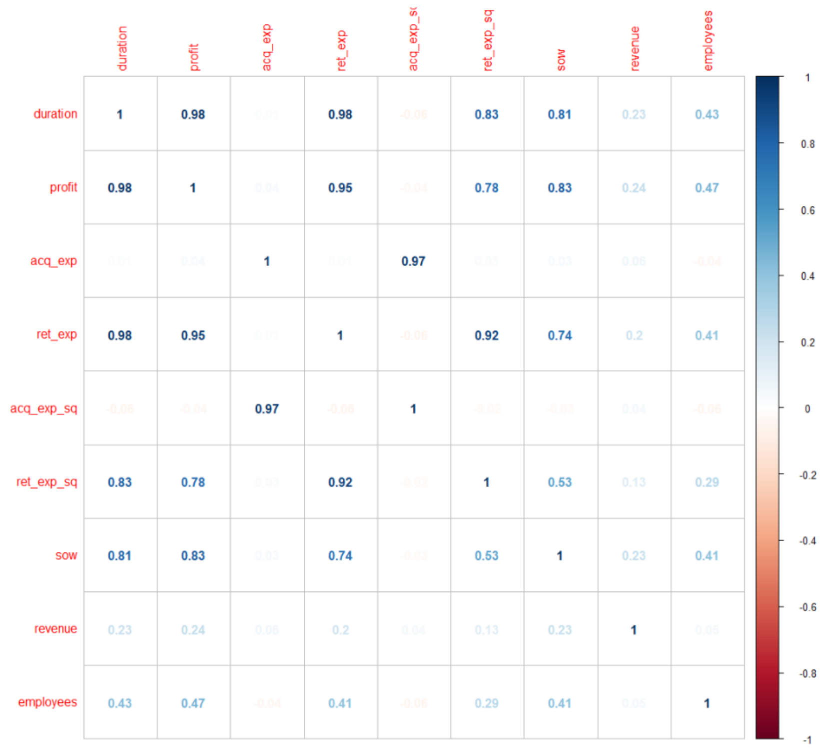
## Review of Related Literature

While still a relatively modern machine learning technique, random forests have seen extensive use since first introduced by Breiman in his paper Random Forests (2001). Specifically, `random forests` have been applied to the problem of predicting customer retention by a handful of researchers. Larivière and Poel compared a model based on `random forests` to traditional prediction models and found it superior to both `logistic` and l `inear regression models` when measured by predictive accuracy (2005). More recently, Sabbeh compared a wide range of popular techniques for customer "churn" prediction in a "battle royal" that tested logistic regression, decision tree, Naïve Bayesian, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), ensemble learning, which includes `random forest` , artificial neural network (ANN), and linear discriminant analysis (LDA). His results indicated that random forest was superior to other techniques with an accuracy of roughly 96%. He did note that SVM was close behind with an accuracy of 94%.

Schaeffer and Sanchez found that the performance of a `random forest` model was superior to an `SVM` model but determined that the accuracy gain was small enough to make SVM a viable option as its performance was much better by several factors (2020). While this difference amounted to only a few seconds in running the model on their dataset, they noted that the additional accuracy of `random forests` may not be worth the cost in time for large datasets. For all other situations where accuracy results in increased profit and time is not a significant factor, the `random forests` technique appears to be emerging as the go-to model for customer retention prediction.

# Methods

The case study aims to accurately predict customer retention and acquisition to target the right customers and decrease the cost of marketing campaigns. To fit the models, we are using decision trees, logistic regression, and random forest. We use the `acquisitionRetention` data set to predict which customers will be acquired. Prior to fitting the models, we performed some data cleaning and looked for missing values and multicollinearity. Additionally, we converted `crossbuy`, `industry`, and acquisition into factors. The correlation plot below shows some multicollinearity between independent variables. For instance `duration` has high correlation with `profit` and `ret_exp`, `profit` has a high correlation with `ret_exp`, and `acq_exp` has a high correlation with `ac_exp_sq`.




Next, we split the data into a 70 percent training set and 30 percent testing set. We fit our first model using a decision tree. This model technique is a supervised learning algorithm that can be used in regression problems. The response variable is `acquisition`, and the independent variables were `acq_exp`, `industry`, `revenue`, and `employees`. Using decision trees has some advantages. One of the advantages of a `Decision Tree` is that the output is easy to interpret.

`Decision Tree` is also helpful in data exploration because it identifies the most significant variables. For instance, the decision tree model indicated that `employees`, `acq_exp`, and `revenue` were more important. Another advantage is that it requires less data cleaning, and outliers have no influence on the model, and it can handle both numerical and categorical variables. In this case study, the response variable, `acquisition`, is categorical. However, decision trees have some disadvantages as well. Overfitting can be a problem when using decision trees. Some techniques can be performed to improve the performance of decision trees like bagging, random forest, and boosting (Arka Roy, DA 6813. DATA ANALYTICS APPLICATIONS SPRING 2021. PowerPoint).

The next technique used was logistic regression. We fit our `glm` model using the following predictors: `acq_exp`, `acq_exp_sq`, `industry`, `revenue`, and `employees`. In the original data, the `acquisition` variable was numerical, with 1 if the prospect was acquired and 0 otherwise. The response variable was converted to a factor to fit a logistic model. Next, we use the `vif()` function to calculate the variance inflation factor to measure how much the variance of a regression coefficient is inflated due to multicollinearity. We eliminated `acq_exp`, which had a $vif > 5$. That left us with a model using the following predictors: `acq_exp_sq`, `industry`, `revenue`, and `employees`. The model's performance increased from 0.7133 accuracy in the decision tree model to 0.8 accuracy on the logistic model. Although the logistic regression model outperformed the decision tree model and identified the variables that were significant to the model, logistic regression models have some limitations. For instance, logistic regression cannot show complex relationships, and multicollinearity can cause issues between independent variables.

The last technique used in this case study was random forest. Random forest is a machine learning method used in regression and classification tasks. Random forest is good at handling missing values, outliers, unbalanced data, and other steps in the data exploration process. Similar to decision trees, random forest is also good at identifying important variables. However, one main disadvantage of random forest is overfitting. To grow our forest, first we decided to fit our forest model by manually selecting the predictors. Our first forest model had the following predictors: `acq_exp`, `acq_exp_sq`, `industry`, and `revenue`. This model gave us an overall error rate of 21.43%.

The next forest model had the following predictors: `acq_exp_sq` , `industry` , and `revenue` . This model gave us an overall error rate of 18.29%. A third forest model was fit with the following predictors: `acq_exp` , `industry` , and `revenue` . And this model gave us an overall error rate of 18.29%, similar to the second model. The next step was to optimize our selection process. To identify the optimal set of hyper-parameters based on `OOB error` , we established a list of possible values for the hyper-parameters, we created a data frame containing all combinations, we created an empty vector to store `OOB error` values, and wrote a for loop. We fitted a new forest using the optimal hyper-parameters with a resulting overall error rate of 17.71%.

Finally, we used the predictions from the random forest hyper-parameters model to create a subset of data where customer where predicted to be acquired. We split this subset data set into training and testing sets and fit a model using random forest. Our response variable for this model was `duration` . We followed the same process as before, first, we made a manual selection of independent variables, and then we optimized our selection, in the same way, explained earlier. The model with the optimized parameters gave us better results (which is presented in the result section).

# Data

The Customer Acquisition dataset is a small portion of 500 customers from a typical B2B firm, where all the customers made their first purchase with the firm at the same time (Kumar & Peterson, 2012). The data is embedded in the `SMCRM` package in R. For this case analysis, we created subsets of the database: a training dataset and a testing dataset. The training dataset contains 350 observations and 14 predicting variables. The testing dataset contains 150 observations and 14 variables. The test dataset is a holdout sample representative of the entire population, and it is used to assess the performance of the model. The variables are as follows:

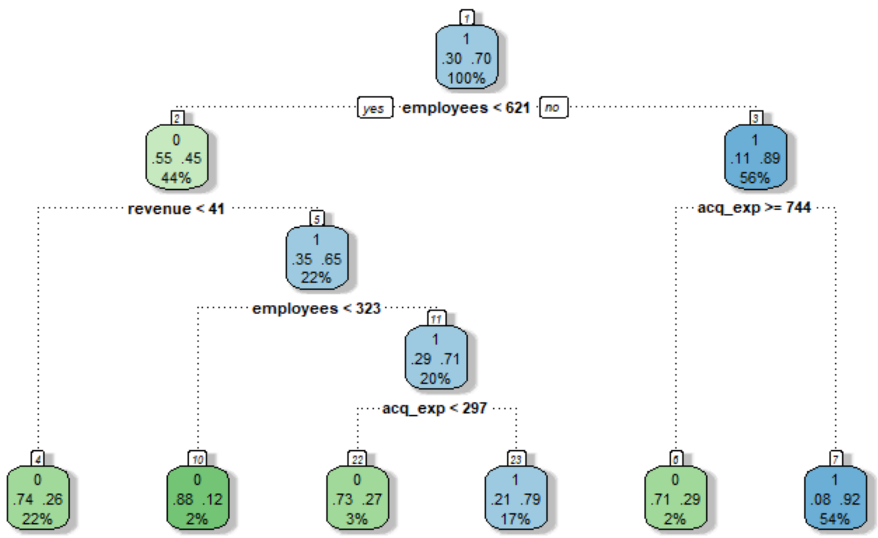
| | Variable Name | Description |
|---|---|---|
| 1. | customer | customers number (from 1 to 500) |
| 2. | acquisition | 1 if the prospect was acquired, 0 otherwise |
| 3. | duration | number of days the customer was a customer of the firm, 0 if acquisition == 0 |
| 4. | profit | customer lifetime value (CLV) of a given customer, -(Acq_Exp) if the customer is not acquired |
| 5. | acq_exp | total dollars spent on trying to acquire this prospect |
| 6. | ret_exp | total dollars spent on trying to retain this customer |
| 7. | acq_exp_sq | square of the total dollars spent on trying to acquire this prospect |
| 8. | ret_exp_sq | square of the total dollars spent on trying to retain this customer |
| 9. | freq | number of purchases the customer made during that customer's lifetime with the firm, 0 if acquisition == 0 |
| 10. | freq_sq | square of the number of purchases the customer made during that customer's lifetime with the firm |
| 11. | crossbuy | number of product categories the customer purchased from during that customer's lifetime with the firm, 0 if acquisition = 0 |
| 12. | sow | Share-of-Wallet; percentage of purchases the customer makes from the given firm given the total amount of purchases across all firms in that category |
| 13. | industry | 1 if the customer is in the B2B industry, 0 otherwise |
| 14. | revenue | annual sales revenue of the prospect's firm (in millions of dollar) |
| 15. | employees | number of employees in the prospect's firm |

Prior to training the model based on the dataset, some data cleaning had to be done. The data preprocessing for this case study entailed looking for missing values, looking at multicollinearity, and possible elimination of irrelevant variables. This dataset was relatively clean and did not have any missing values. To better understand and make inferences about the data, some numerical and graphical summaries were performed. The Correlation Matrix and Correlation Plot were utilized to assess any variables that had a correlationship with each other. It revealed that `profit` , and `ret_exp` , `acq_exp` , and `acq_exp_sq` , `ret_exp` , and `ret_exp_sq` , `freq` , and `freq_sq` , had a strong correlation.

Additionally, while creating our models, we did not include some variables that were not relevant for our analysis. We used `as. Factor()` to convert `industry` , `crossbuy` , and `Acquisition` into factor variables for Decision Tree, Logistic Regression, and Random Forest Models.

# Results

For the initial model selection of acquisition, when constructing the tree model, the only variables included were `acq_exp` , `industry` , `revenue` and `employees` . Other variables were excluded due to providing perfect separation for the acquisition variable. For example, the `duration` variable is equal to zero for all customers that were not acquired. As a result, the `duration` variable does not allow for deeper insight of other variables' effects, as we would only have to determine if a customer's duration is greater than zero to know if they were acquired or not.

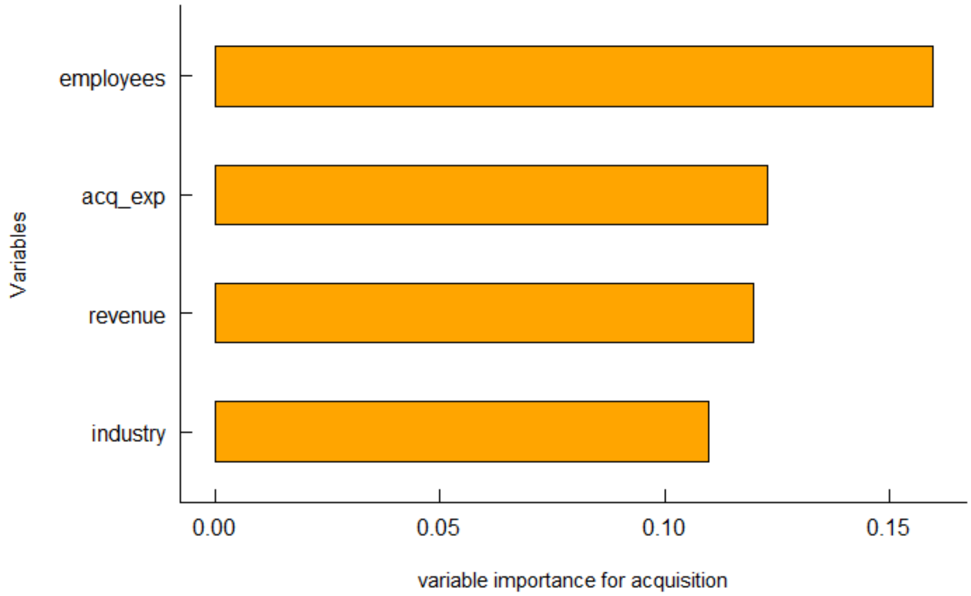



The resulting tree model used an initial split at `employees` $< 621$, with further splits at `revenue` , `acq_exp` , and `employees` , once more. The accuracy of the model was roughly 71.33%, with 73.58% sensitivity and 65.91% specificity.

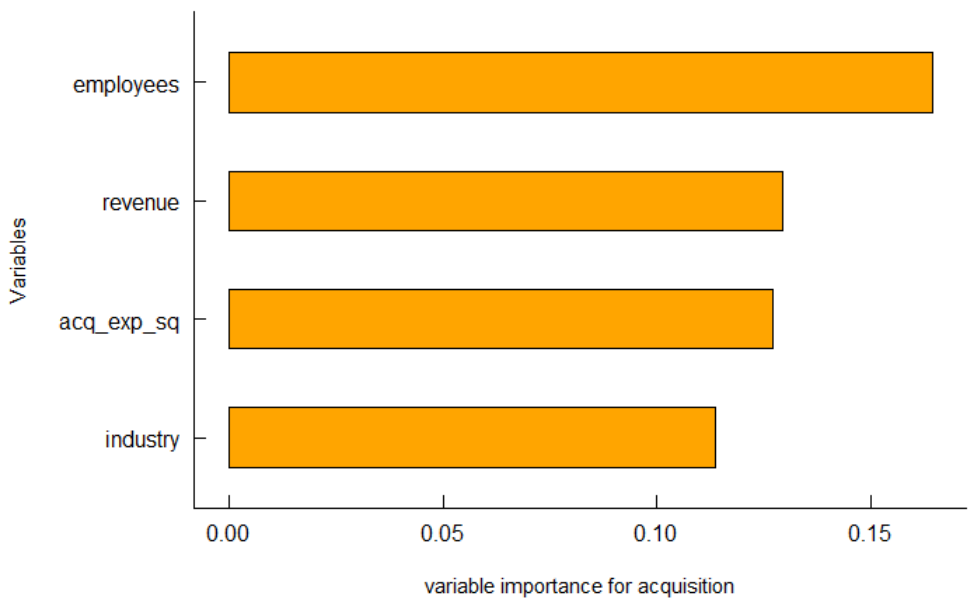
A similar approach was used when constructing a logistic model for the data. Initially the variables used were the same as the tree model, with the addition of the `acq_exp_sq` . After running the model, a high correlation was found between the `acq_exp` and `acq_exp_sq` , the `acq_exp` was removed from the model and the model was rerun. The resulting model showed that the variables `industry` , `revenue` , and `employees` all had significant effects on customer acquisition. The model displayed an overall accuracy of 80%, with sensitivity of 77.88% and specificity of 86.49%.

For the forest model, two different approaches were taken. The first approach was a manual selection, while the second approach used an optimized selection to determine which hyper parameters were appropriate for the model.

The manual model created used the same variables as the initial tree model as an input formula. The resulting metrics showed an accuracy of roughly 81.71% with a sensitivity of 91.02% and a 59.05% specificity. This model showed a large improvement over the previous two models created. Additionally, the `employees` variable showed the most importance of all variables, with roughly 0.0597 importance overall. `acq_exp` and `revenue` showed similar importance, 0.0229 and 0.0199 respectively. Finally, the `industry` variable had the least amount of importance for acquisition of a customer. The graph below shows these metrics with a constant of 0.1 added for easier viewing.



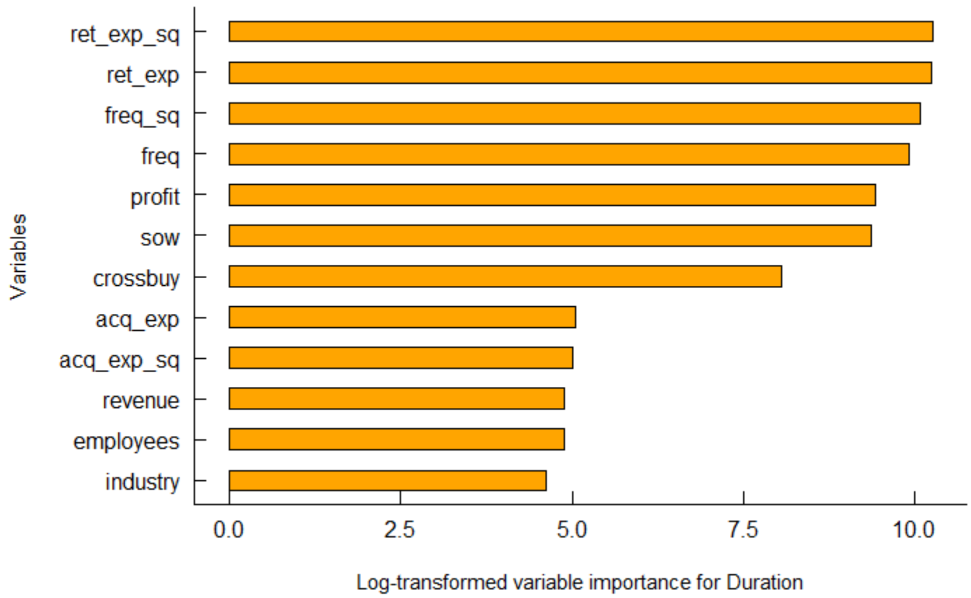

The optimized model was run with `mtry` , the number of variables randomly selected for splitting a node, equal to 1, a minimum terminal node size of 1, and 1000 trees. The results show an accuracy of 82.29%, 55.24% specificity, and a 99.94% sensitivity, a further improvement from the manually selected random forest model. The importance values for `employees` , `acq_exp_sq` , `revenue` , and `industry` were 0.0644, 0.0271, 0.0265, and 0.0135, respectively. The graphic again shows the values with a constant of 0.1 added for ease of viewing. Although the model showed average results for properly predicting negative cases, it displayed the best overall accuracy, as well as having excellent ability to predict positive cases.



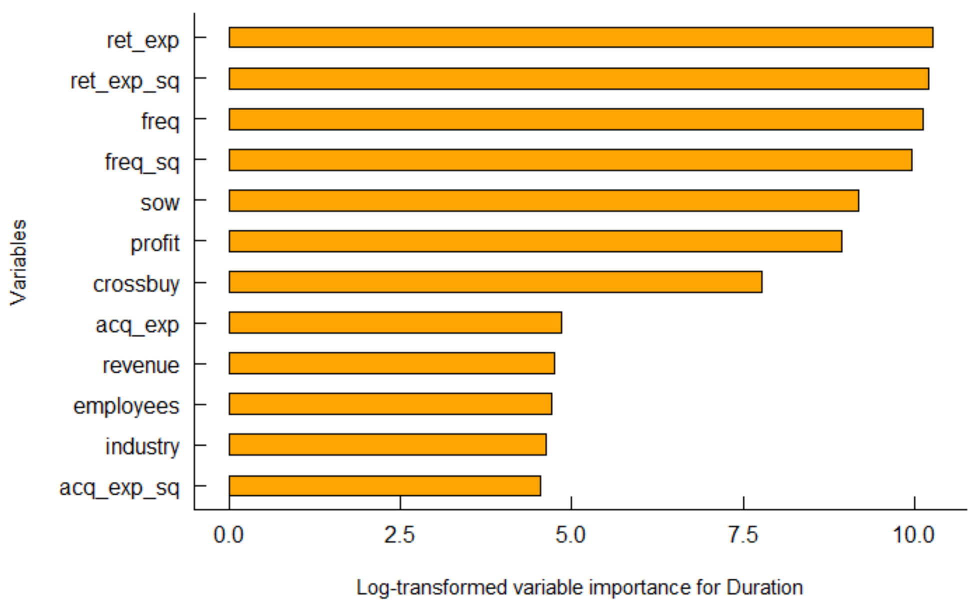

Predictions were then carried out using this model on the complete dataset. From these predictions, data points with a positive prediction of acquisition were separated from the dataset. The average `duration` for this subsetted data was roughly 949.32 days. A random forest model with manual selection and model with optimized hyper parameters was then created with duration as the response variable, including all variables except for acquisition. The model with manual selection produced an error rate of 1404.75. The variables with the greatest importance are `ret_exp_sq` , `ret_exp` , `freq_sq` , and `freq` .

```{r}
summary(Acquired.df$duration)
```

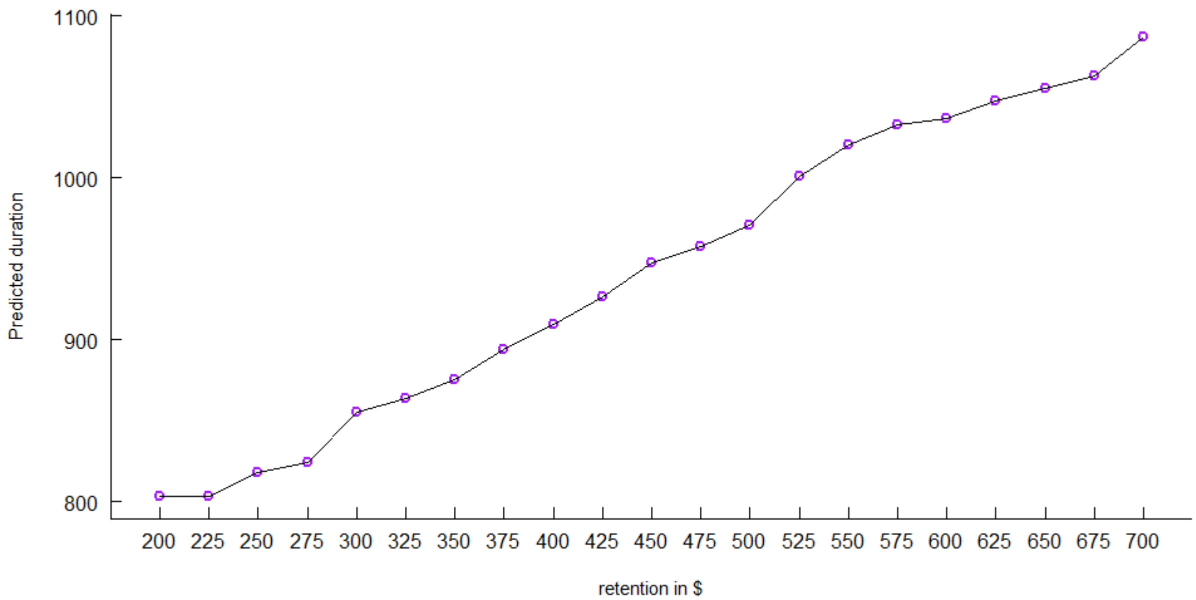
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    0.0   869.2  1038.0   949.3  1207.0  1673.0 
```

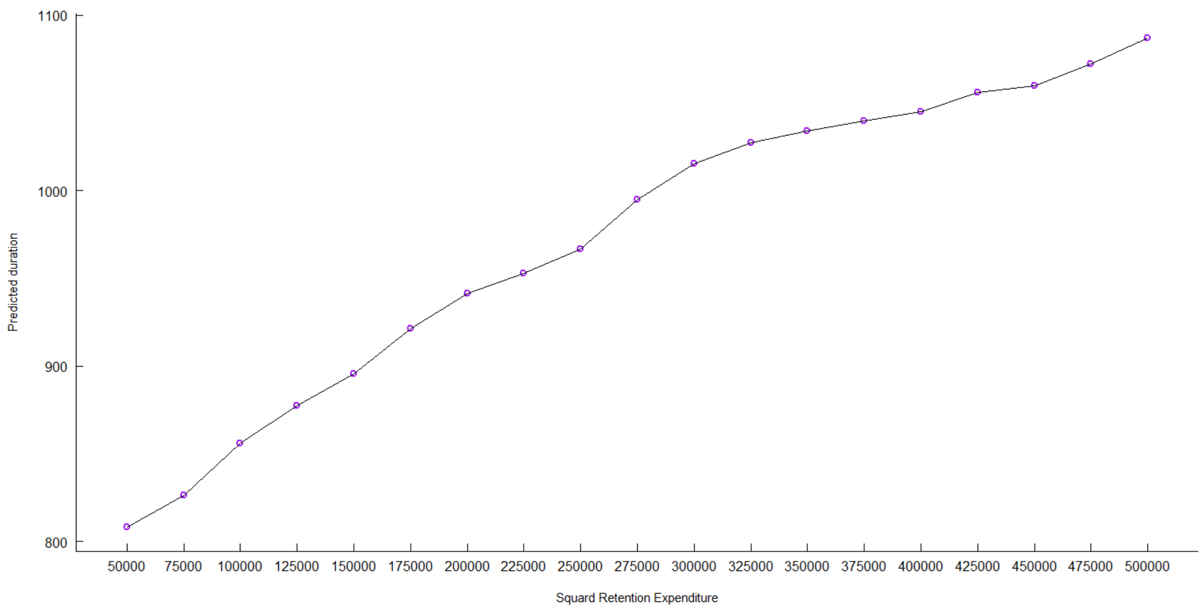


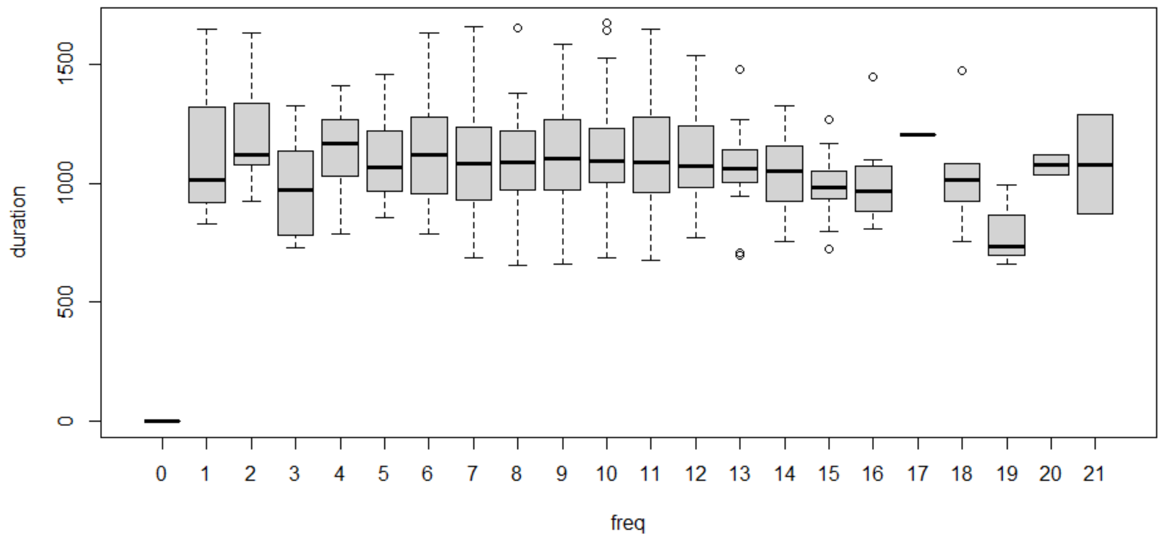

The same variables were used as the input formulas to optimize the hyper parameters for the second random forest model. The resulting optimized hyper parameters were an `mtry` of 6, minimum terminal node size of 1, and 1000 trees. The optimized model returned an error rate of 1033.72, with a mean square error of 1525.854 and a mean absolute error of 25.368. The variables with greatest importance were the `ret_exp_sq` , `ret_exp` , `freq_sq` , and `freq` variables. Interactions between these four variables were inspected, and the model rerun with these interaction variables. However, the new rerun model generated the same optimized hyperparameters and the same results as the model that did not include interactions.

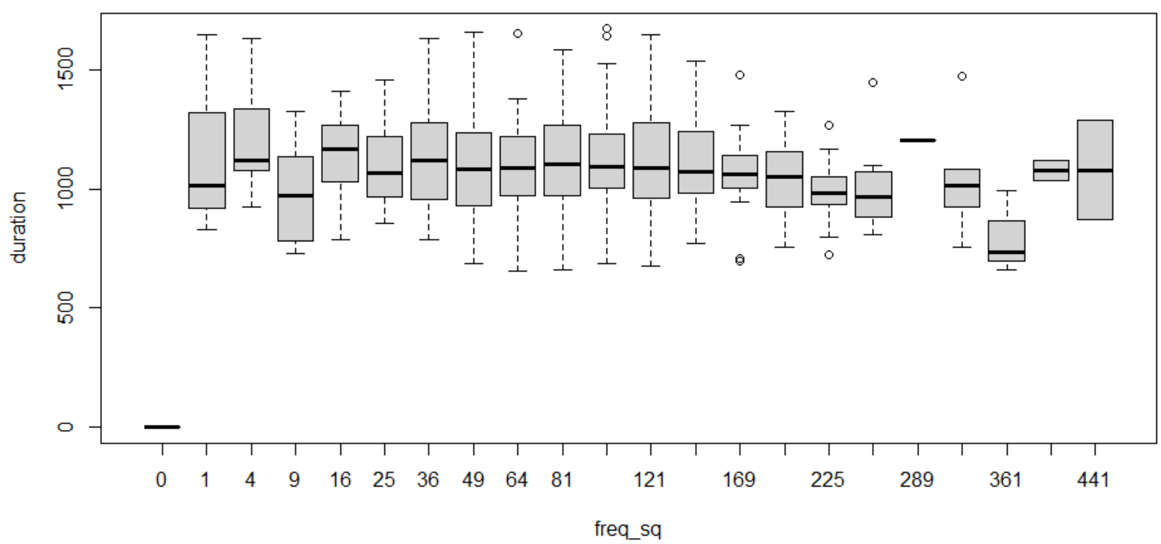



Partial dependence plots were then created for each of the variables with the greatest importance. From these plots, it can be observed that there are no linear relations between the variables and the duration of an acquired customer.


1100
1000
900
800
Predicted duration
200 225 250 275 300 325 350 375 400 425 450 475 500 525 550 575 600 625 650 675 700
retention in $


1100
1000
900
800
Predicted duration
50000 75000 100000 125000 150000 175000 200000 225000 250000 275000 300000 325000 350000 375000 400000 425000 450000 475000 500000
Squard Retention Expenditure


duration
1500
1000
500
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
freq


duration
1500
1000
500
0
0 1 4 9 16 25 36 49 64 81 121 169 225 289 361 441
freq_sq

# Conclusion & Recommendations

In conclusion, a tuned Random Forest model outperformed both logistic and decision tree models in predicting new customer acquisition. The variables that significantly impact customer acquisition include `employees` , `acq_exp_sq` , `revenue` , and `industry` . It is possible that more and more customers are interested in a better quality of service, it makes sense that the number of employees has the most significant effect on customer acquisition, followed by the dollars invested in attaining the customer.

Based on our final results, we identified that the `ret_exp_sq` , `ret_exp` , `freq_sq` , and `freq` variables significantly impact the customer `duration` variable. Additionally, we determined that the average duration of the customer is around 949.32 days, i.e., 2.6 years.

The company would gain insightful information if, in addition to creating predictive models to determine the significant variables impacting customer acquisition and retention, it also analyzed customer demographics to understand the customer base better. This knowledge will further allow the company to deploy more efficient and targeted approaches to utilize scarce resources to gain business while reducing the cost of marketing campaigns and increasing profitability.

# References

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32

Kumar, V., & Peterson, J. (2012). Statistical Methods in Customer Relationship Management. Wiley.

Larivière, B., & Van den Poel, D. (2005). Predicting customer retention and profitability by using random forests and regression forests techniques. *Expert Systems with Applications*, 29(2), 472–484.

Schaeffer, S., & Rodriguez Sanchez, S. (2020). Forecasting client retention — A machine-learning approach. *Journal of Retailing and Consumer Services*, 52, 101918–.

Sabbeh, S. (2018). Machine-Learning Techniques for Customer Retention: A Comparative Study. *International Journal of Advanced Computer Science and Applications*, 9(2), 273-281.

# Data Exploration

## Data Importing

```
data("acquisitionRetention")
```

```
View(acquisitionRetention)
str(acquisitionRetention)
```

```
## 'data.frame':    500 obs. of  15 variables:
##  $ customer   : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ acquisition: num  1 1 1 0 1 1 1 1 0 0 ...
##  $ duration   : num  1635 1039 1288 0 1631 ...
##  $ profit     : num  6134 3524 4081 -638 5446 ...
##  $ acq_exp    : num  694 460 249 638 589 ...
##  $ ret_exp    : num  972 450 805 0 920 ...
##  $ acq_exp_sq : num  480998 211628 62016 407644 346897 ...
##  $ ret_exp_sq : num  943929 202077 648089 0 846106 ...
##  $ freq       : num  6 11 21 0 2 7 15 13 0 0 ...
##  $ freq_sq    : num  36 121 441 0 4 49 225 169 0 0 ...
##  $ crossbuy   : num  5 6 6 0 9 4 5 5 0 0 ...
##  $ sow        : num  95 22 90 0 80 48 51 23 0 0 ...
##  $ industry   : num  1 0 0 0 0 1 0 1 0 1 ...
##  $ revenue    : num  47.2 45.1 29.1 40.6 48.7 ...
##  $ employees  : num  898 686 1423 181 631 ...
```

## Data Cleaning

```
sapply(acquisitionRetention, function(x) sum(is.na(x)))
```

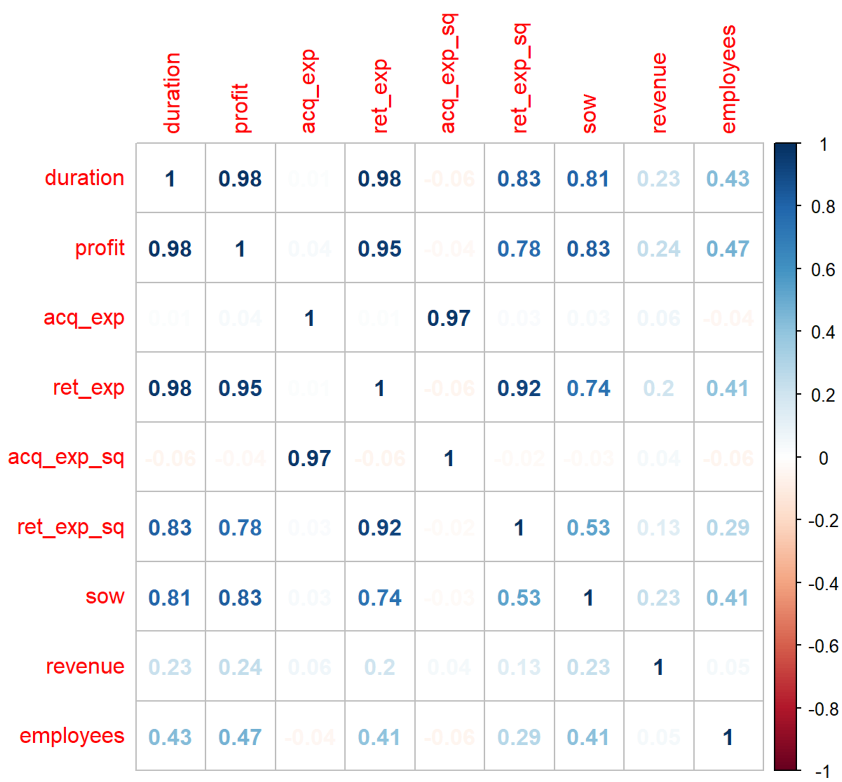
```
##    customer acquisition    duration      profit     acq_exp     ret_exp
##           0           0           0           0           0           0
##  acq_exp_sq  ret_exp_sq        freq     freq_sq    crossbuy         sow
##           0           0           0           0           0           0
##    industry     revenue   employees
##           0           0           0
```

```
acquisitionRetention <- acquisitionRetention[,2:15]
acquisitionRetention$crossbuy <- as.factor(acquisitionRetention$crossbuy)
acquisitionRetention$industry <- as.factor(acquisitionRetention$industry)
acquisitionRetention$acquisition <- as.factor(acquisitionRetention$acquisition)
str(acquisitionRetention)
```

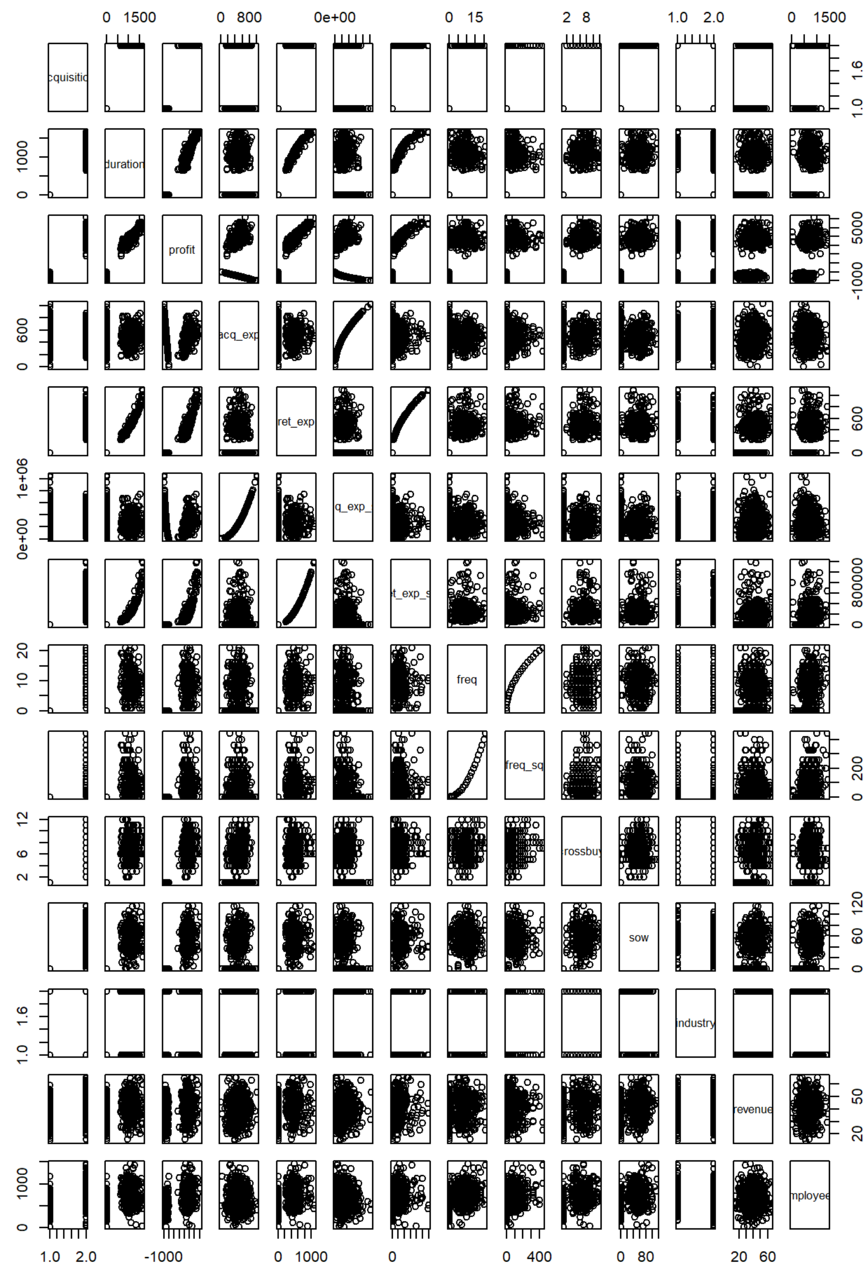
```
## 'data.frame':    500 obs. of  14 variables:
##  $ acquisition: Factor w/ 2 levels "0","1": 2 2 2 1 2 2 2 2 1 1 ...
##  $ duration   : num  1635 1039 1288 0 1631 ...
##  $ profit     : num  6134 3524 4081 -638 5446 ...
##  $ acq_exp    : num  694 460 249 638 589 ...
##  $ ret_exp    : num  972 450 805 0 920 ...
##  $ acq_exp_sq : num  480998 211628 62016 407644 346897 ...
##  $ ret_exp_sq : num  943929 202077 648089 0 846106 ...
##  $ freq       : num  6 11 21 0 2 7 15 13 0 0 ...
##  $ freq_sq    : num  36 121 441 0 4 49 225 169 0 0 ...
##  $ crossbuy   : Factor w/ 12 levels "0","1","2","3",..: 6 7 7 1 10 5 6 6 1 1 ...
##  $ sow        : num  95 22 90 0 80 48 51 23 0 0 ...
##  $ industry   : Factor w/ 2 levels "0","1": 2 1 1 1 1 2 1 2 1 2 ...
##  $ revenue    : num  47.2 45.1 29.1 40.6 48.7 ...
##  $ employees  : num  898 686 1423 181 631 ...
```

# Data Correlations

```
corrplot(cor(acquisitionRetention[,c(2:7,11,13:14)]), method = "number")
```




```
pairs(acquisitionRetention)
```




# Data Splitting

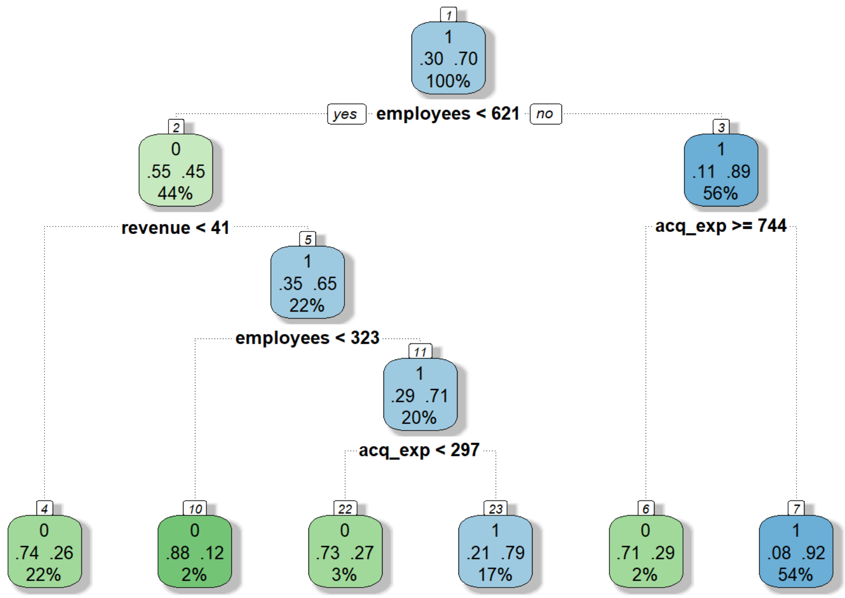
## Split for Acquisiton prediction

```
set.seed(123)
idx.train <- sample(1:nrow(acquisitionRetention), size = 0.7 * nrow(acquisitionRetention))
train.df <- acquisitionRetention[idx.train,]
test.df <- acquisitionRetention[-idx.train,]
```

# Model Creation & Predictions

# Tree Model - Acquisition

```
set.seed(123)
dt.model <- rpart(acquisition ~ acq_exp + industry + revenue + employees, data = train.df) # simple DT model

rattle::fancyRpartPlot(dt.model, sub = "") # vizualize the DT
```




```
predicted.acquisition <- predict(dt.model, newdata = test.df, type = "class")
View(predicted.acquisition)
```

```
caret::confusionMatrix(as.factor(test.df$acquisition),as.factor(predicted.acquisition), positive='1')
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction  0  1
##          0 29 28
##          1 15 78
##                                           
##                Accuracy : 0.7133          
##                  95% CI : (0.6339, 0.7841)
##     No Information Rate : 0.7067          
##     P-Value [Acc > NIR] : 0.46919         
##                                           
##                   Kappa : 0.3635          
##                                           
##  Mcnemar's Test P-Value : 0.06725         
##                                           
##             Sensitivity : 0.7358          
##             Specificity : 0.6591          
##          Pos Pred Value : 0.8387          
##          Neg Pred Value : 0.5088          
##              Prevalence : 0.7067          
##          Detection Rate : 0.5200          
##    Detection Prevalence : 0.6200          
##       Balanced Accuracy : 0.6975          
##                                           
##        'Positive' Class : 1               
## 
```

# Logistic Model - Acquisition

```
set.seed(123)
glm.model <- glm(acquisition ~ acq_exp + acq_exp_sq + industry + revenue + employees, data = train.df, family = "bino
mial")
summary(glm.model)
```

```
## 
## Call:
## glm(formula = acquisition ~ acq_exp + acq_exp_sq + industry + 
##     revenue + employees, family = "binomial", data = train.df)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -2.2237  -0.2916   0.1958   0.5023   2.4956  
## 
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -1.524e+01  1.879e+00  -8.108 5.15e-16 ***
## acq_exp      3.286e-02  5.336e-03   6.158 7.39e-10 ***
## acq_exp_sq  -3.267e-05  5.334e-06  -6.124 9.15e-10 ***
## industry1    1.695e+00  3.585e-01   4.728 2.27e-06 ***
## revenue      8.734e-02  1.784e-02   4.895 9.84e-07 ***
## employees    7.480e-03  1.011e-03   7.399 1.37e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 427.61  on 349  degrees of freedom
## Residual deviance: 231.92  on 344  degrees of freedom
## AIC: 243.92
## 
## Number of Fisher Scoring iterations: 6
```

```
car::vif(glm.model)
```

```
##    acq_exp acq_exp_sq   industry    revenue  employees
##  28.382762  28.030374   1.129658   1.082911   1.265077
```

```
set.seed(123)
glm.model2 <- glm(acquisition ~ acq_exp_sq + industry + revenue + employees, data = train.df, family = "binomial")
summary(glm.model2)
```

```
##
## Call:
## glm(formula = acquisition ~ acq_exp_sq + industry + revenue +
##     employees, family = "binomial", data = train.df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1886  -0.5368   0.3057   0.6448   2.4065
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.097e+00  9.819e-01  -7.228 4.91e-13 ***
## acq_exp_sq  -6.132e-07  8.590e-07  -0.714    0.475
## industry1    1.303e+00  3.092e-01   4.215 2.50e-05 ***
## revenue      8.807e-02  1.623e-02   5.425 5.80e-08 ***
## employees    6.561e-03  8.679e-04   7.559 4.06e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 427.61  on 349  degrees of freedom
## Residual deviance: 285.96  on 345  degrees of freedom
## AIC: 295.96
##
## Number of Fisher Scoring iterations: 5
```

```
car::vif(glm.model2)
```

```
## acq_exp_sq   industry    revenue  employees
##   1.018414   1.087649   1.089387   1.120444
```

```
glm.preds <- predict(glm.model2, newdata = test.df, type = "response")
test.df$PredChoice = ifelse(glm.preds >= 0.5, 1,0)
test.df$PredChoice = as.factor(test.df$PredChoice)
```

```
caret::confusionMatrix(as.factor(test.df$acquisition),as.factor(test.df$PredChoice), positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 32 25
##          1  5 88
##
##                Accuracy : 0.8
##                  95% CI : (0.727, 0.8608)
##     No Information Rate : 0.7533
##     P-Value [Acc > NIR] : 0.1073527
##
##                   Kappa : 0.5446
##
##  Mcnemar's Test P-Value : 0.0005226
##
##             Sensitivity : 0.7788
##             Specificity : 0.8649
##          Pos Pred Value : 0.9462
##          Neg Pred Value : 0.5614
##              Prevalence : 0.7533
##          Detection Rate : 0.5867
##    Detection Prevalence : 0.6200
##       Balanced Accuracy : 0.8218
##
##        'Positive' Class : 1
##
```

# Forest Model - Acquisition

## theme for nice plotting

```
theme_nice <- theme_classic()+
                theme(
                  axis.line.y.left = element_line(colour = "black"),
                  axis.line.y.right = element_line(colour = "black"),
                  axis.line.x.bottom = element_line(colour = "black"),
                  axis.line.x.top = element_line(colour = "black"),
                  axis.text.y = element_text(colour = "black", size = 12),
                  axis.text.x = element_text(color = "black", size = 12),
                  axis.ticks = element_line(color = "black")) +
                theme(
                  axis.ticks.length = unit(-0.25, "cm"),
                  axis.text.x = element_text(margin=unit(c(0.5,0.5,0.5,0.5), "cm")),
                  axis.text.y = element_text(margin=unit(c(0.5,0.5,0.5,0.5), "cm")))
```

## Manual Selection

```
set.seed(123)
forest1 <- rfsrc(acquisition ~ acq_exp + acq_exp_sq + industry + revenue + employees,
                            data = train.df,
                            importance = TRUE,
                            ntree = 1000)

forest1
```

```
##                          Sample size: 350
##            Frequency of class labels: 105, 245
##                      Number of trees: 1000
##            Forest terminal node size: 1
##        Average no. of terminal nodes: 49.886
## No. of variables tried at each split: 3
##               Total no. of variables: 5
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 221
##                             Analysis: RF-C
##                               Family: class
##                       Splitting rule: gini *random*
##        Number of random split points: 10
##               Normalized brier score: 56.97
##                                  AUC: 85.2
##                           Error rate: 0.21, 0.43, 0.12
##
## Confusion matrix:
##
##           predicted
##   observed  0   1 class.error
##          0 60  45      0.4286
##          1 30 215      0.1224
##
##  Overall error rate: 21.43%
```

```
set.seed(123)
forest2 <- rfsrc(acquisition ~ acq_exp_sq + industry + revenue + employees,
                            data = train.df,
                            importance = TRUE,
                            ntree = 1000)

forest2
```

```
##                          Sample size: 350
##            Frequency of class labels: 105, 245
##                      Number of trees: 1000
##            Forest terminal node size: 1
##        Average no. of terminal nodes: 53.859
## No. of variables tried at each split: 2
##               Total no. of variables: 4
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 221
##                             Analysis: RF-C
##                               Family: class
##                       Splitting rule: gini *random*
##        Number of random split points: 10
##               Normalized brier score: 55.68
##                                  AUC: 85.55
##                           Error rate: 0.18, 0.41, 0.09
##
## Confusion matrix:
##
##           predicted
##   observed  0   1 class.error
##          0 62  43      0.4095
##          1 22 223      0.0898
##
##  Overall error rate: 18.29%
```

```r
set.seed(123)
forest3 <- rfsrc(acquisition ~ acq_exp + industry + revenue + employees,
                            data = train.df,
                            importance = TRUE,
                            ntree = 1000)

forest3
```

```
##                          Sample size: 350
##            Frequency of class labels: 105, 245
##                      Number of trees: 1000
##            Forest terminal node size: 1
##        Average no. of terminal nodes: 53.859
## No. of variables tried at each split: 2
##               Total no. of variables: 4
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 221
##                             Analysis: RF-C
##                               Family: class
##                       Splitting rule: gini *random*
##        Number of random split points: 10
##               Normalized brier score: 55.68
##                                  AUC: 85.55
##                           Error rate: 0.18, 0.41, 0.09
##
## Confusion matrix:
##
##           predicted
##   observed  0   1 class.error
##          0 62  43      0.4095
##          1 22 223      0.0898
##
##  Overall error rate: 18.29%
```

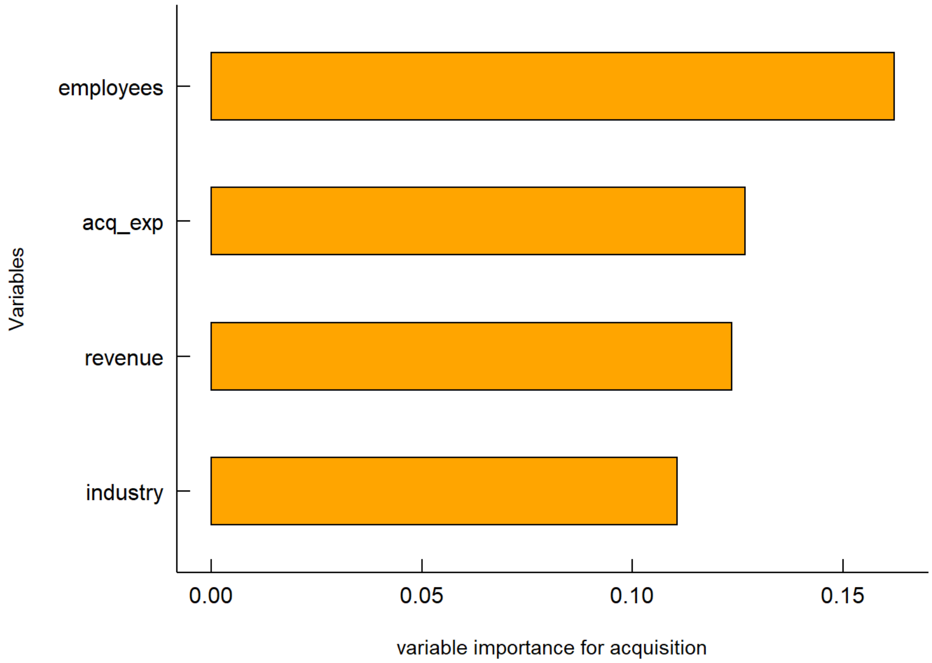
```r
forest3$importance
```

```
##                  all          0            1
## acq_exp   0.02663842 0.27415814 -0.015089238
## industry  0.01059101 0.06006108  0.014978288
## revenue   0.02355819 0.19235080  0.008210321
## employees 0.06210708 0.55219954  0.002108055
```

```r
forest3$importance[,1]
```

```
##    acq_exp   industry    revenue  employees
## 0.02663842 0.01059101 0.02355819 0.06210708
```

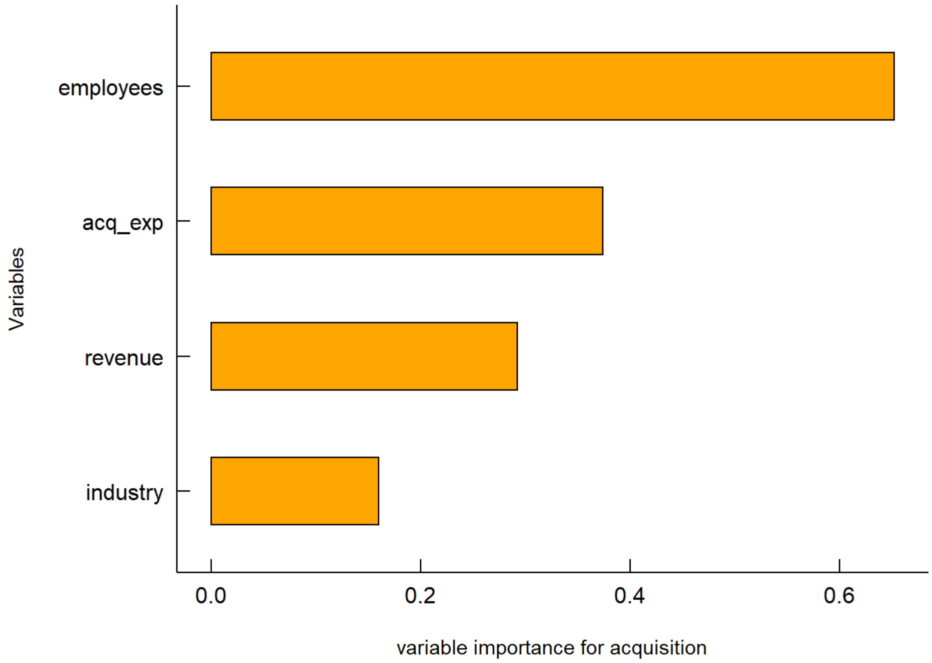
```r
data.frame(importance = forest3$importance[,1] +.1) %>% # add a large +ve constant

  tibble::rownames_to_column(var = "variable") %>%
  ggplot(aes(x = reorder(variable,importance), y = importance)) +
    geom_bar(stat = "identity", fill = "orange", color = "black", width = 0.5)+
    coord_flip() +
    labs(x = "Variables", y = "variable importance for acquisition") +
    theme_nice
```




```
forest3$importance[,2]
```

```
##    acq_exp   industry    revenue  employees
## 0.27415814 0.06006108 0.19235080 0.55219954
```

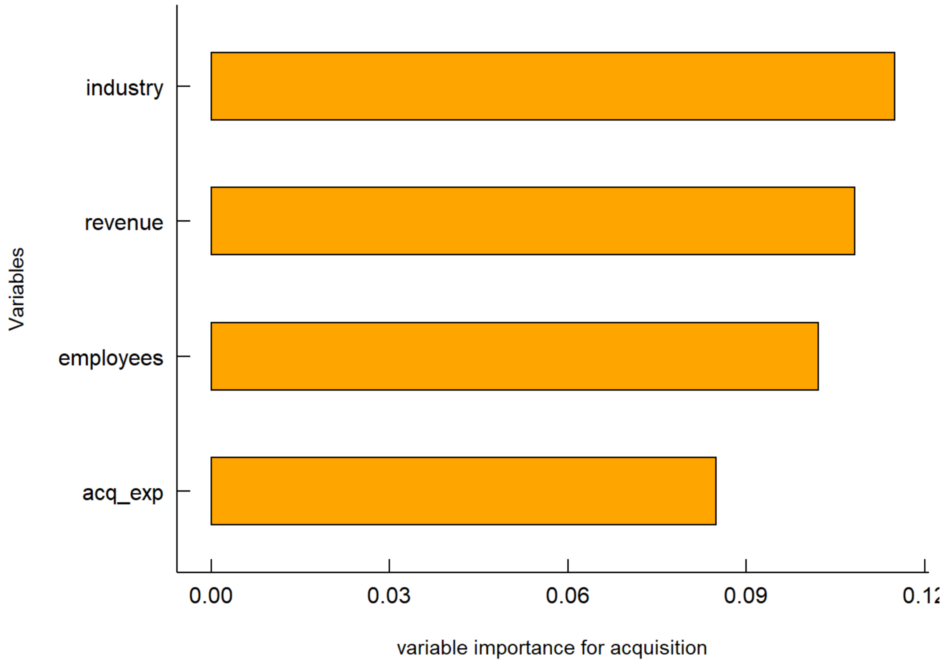
```
data.frame(importance = forest3$importance[,2] +.1) %>% # add a large +ve constant

  tibble::rownames_to_column(var = "variable") %>%
  ggplot(aes(x = reorder(variable,importance), y = importance)) +
    geom_bar(stat = "identity", fill = "orange", color = "black", width = 0.5)+
    coord_flip() +
    labs(x = "Variables", y = "variable importance for acquisition") +
    theme_nice
```




```
forest3$importance[,3]
```

```
##      acq_exp     industry      revenue    employees
## -0.015089238  0.014978288  0.008210321  0.002108055
```

```
data.frame(importance = forest3$importance[,3] +.1) %>% # add a large +ve constant

  tibble::rownames_to_column(var = "variable") %>%
  ggplot(aes(x = reorder(variable,importance), y = importance)) +
    geom_bar(stat = "identity", fill = "orange", color = "black", width = 0.5)+
    coord_flip() +
    labs(x = "Variables", y = "variable importance for acquisition") +
    theme_nice
```




## Optimized Selection

```
# Establish a list of possible values for hyper-parameters
mtry.values <- seq(1,4,1)
nodesize.values <- seq(1,4,1)
ntree.values <- seq(1e3,6e3,1e3)

# Create a data frame containing all combinations
hyper_grid <- expand.grid(mtry = mtry.values, nodesize = nodesize.values, ntree = ntree.values)

# Create an empty vector to store OOB error values
oob_err <- c()

# Write a loop over the rows of hyper_grid to train the grid of modelsfor (i in 1:nrow(hyper_grid)) {
for (i in 1:nrow(hyper_grid)) {
    # Train a Random Forest model
   set.seed(100)
   model <- rfsrc(acquisition ~ acq_exp_sq + industry + revenue + employees,
                        data = train.df,
                        mtry = hyper_grid$mtry[i],
                        nodesize = hyper_grid$nodesize[i],
                        ntree = hyper_grid$ntree[i])


    # Store OOB error for the model
    oob_err[i] <- model$err.rate[length(model$err.rate)]
}

# Identify optimal set of hyperparmeters based on OOB error
opt_i <- which.min(oob_err)
print(hyper_grid[opt_i,])
```

```
##   mtry nodesize ntree
## 1    1        1  1000
```

```r
set.seed(111)
forest.hyper <- rfsrc(acquisition ~ acq_exp_sq + industry + revenue + employees,
                            data = train.df,
                            mtry = 1,
                            nodesize = 1,
                            ntree = 1000,
                            importance = TRUE)
forest.hyper
```

```
##                          Sample size: 350
##            Frequency of class labels: 105, 245
##                      Number of trees: 1000
##            Forest terminal node size: 1
##        Average no. of terminal nodes: 33.097
## No. of variables tried at each split: 1
##               Total no. of variables: 4
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 221
##                             Analysis: RF-C
##                               Family: class
##                       Splitting rule: gini *random*
##        Number of random split points: 10
##               Normalized brier score: 54.08
##                                  AUC: 87.41
##                           Error rate: 0.18, 0.45, 0.06
##
## Confusion matrix:
##
##           predicted
##   observed  0   1 class.error
##          0 58  47      0.4476
##          1 15 230      0.0612
##
##  Overall error rate: 17.71%
```

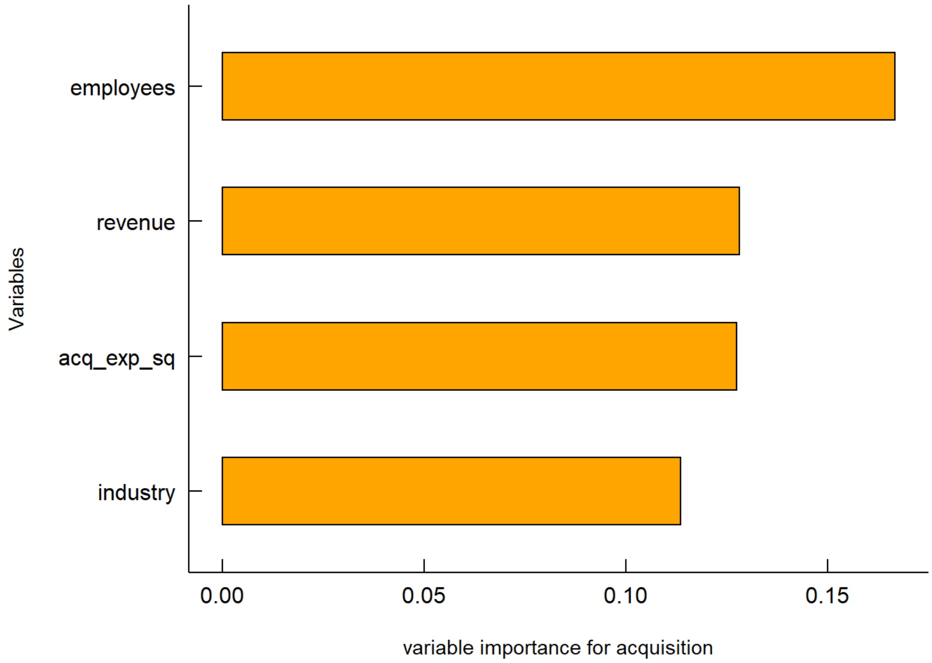
```r
forest.hyper$importance
```

```
##                   all          0          1
## acq_exp_sq 0.02751073 0.21487371 0.01364688
## industry   0.01357054 0.08620837 0.01520019
## revenue    0.02810808 0.20218839 0.02141340
## employees  0.06665774 0.49032627 0.04615532
```

```r
forest.hyper$importance[,1]
```

```
## acq_exp_sq   industry    revenue  employees
## 0.02751073 0.01357054 0.02810808 0.06665774
```

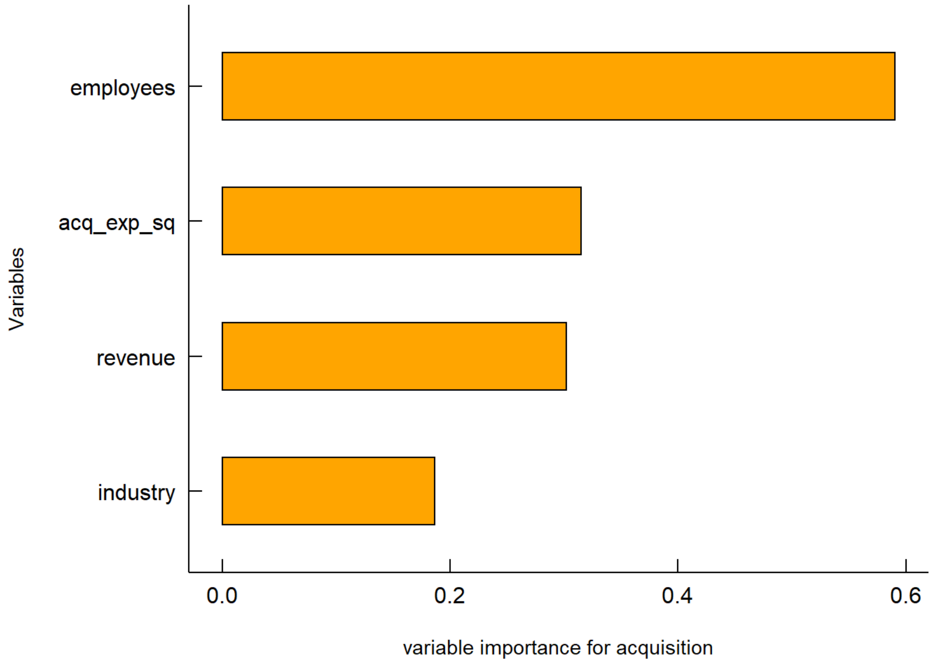
```r
data.frame(importance = forest.hyper$importance[,1] +.1) %>% # add a large +ve constant

  tibble::rownames_to_column(var = "variable") %>%
  ggplot(aes(x = reorder(variable,importance), y = importance)) +
    geom_bar(stat = "identity", fill = "orange", color = "black", width = 0.5)+
    coord_flip() +
    labs(x = "Variables", y = "variable importance for acquisition") +
    theme_nice
```




```
forest.hyper$importance[,2]
```

```
## acq_exp_sq   industry    revenue  employees 
## 0.21487371 0.08620837 0.20218839 0.49032627
```

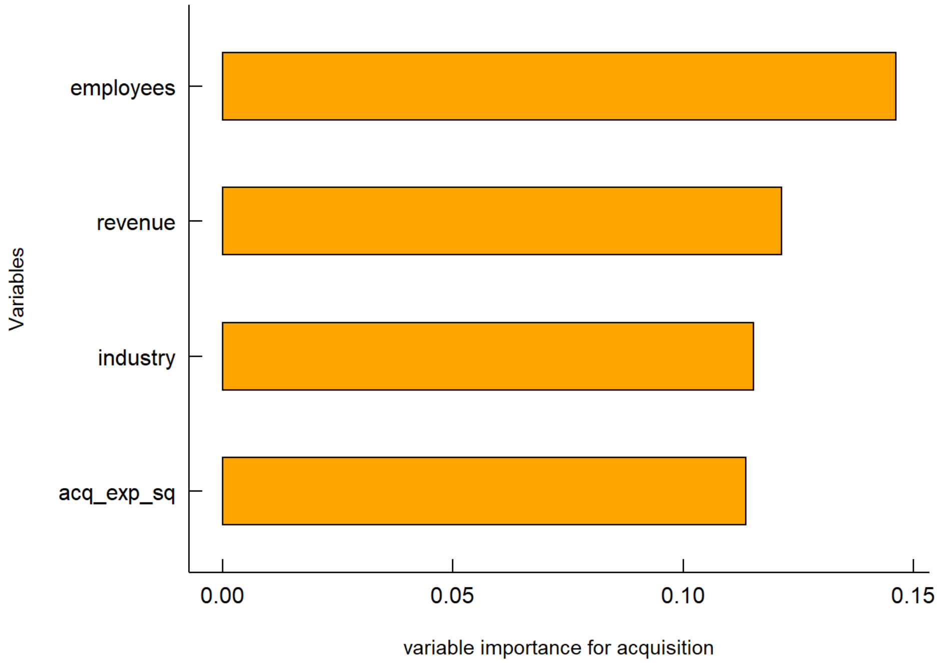
```
data.frame(importance = forest.hyper$importance[,2] + .1) %>% # add a large +ve constant

  tibble::rownames_to_column(var = "variable") %>%
  ggplot(aes(x = reorder(variable,importance), y = importance)) +
    geom_bar(stat = "identity", fill = "orange", color = "black", width = 0.5)+
    coord_flip() +
    labs(x = "Variables", y = "variable importance for acquisition") +
    theme_nice
```




```
forest.hyper$importance[,3]
```

```
## acq_exp_sq   industry    revenue  employees
## 0.01364688 0.01520019 0.02141340 0.04615532
```

```
data.frame(importance = forest.hyper$importance[,3] + .1) %>% # add a large +ve constant

  tibble::rownames_to_column(var = "variable") %>%
  ggplot(aes(x = reorder(variable,importance), y = importance)) +
    geom_bar(stat = "identity", fill = "orange", color = "black", width = 0.5)+
    coord_flip() +
    labs(x = "Variables", y = "variable importance for acquisition") +
    theme_nice
```




```
error.df <- data.frame(pred1 = predict.rfsrc(forest3,newdata = test.df)$class,
             pred2 = predict.rfsrc(forest.hyper, newdata = test.df)$class,
             actual = test.df$acquisition)
```

```
PredsAll = predict.rfsrc(forest.hyper,newdata = acquisitionRetention)$class

Acquisition2.df <- cbind(acquisitionRetention,PredsAll)
```

```
Acquired.df <- filter(Acquisition2.df, PredsAll == "1")
```

## Split for Duration prediction

```
set.seed(123)
idx.train_1 <- sample(1:nrow(Acquired.df), size = 0.7 * nrow(Acquired.df))
acq_train.df <- Acquired.df[idx.train_1,]
acq_test.df <- Acquired.df[-idx.train_1,]
```

```
mean(Acquired.df$duration)
```

```
## [1] 949.3203
```

# Forest Duration Model

## Manual Selection

```
set.seed(123)
forest_duration <- rfsrc(duration ~ profit + acq_exp + acq_exp_sq + ret_exp + ret_exp_sq + freq + freq_sq + crossbuy
 + sow + industry + revenue +employees,
                            data = acq_train.df,
                            importance = TRUE,
                            ntree = 1000)

forest_duration
```

```
##                          Sample size: 268
##                      Number of trees: 1000
##            Forest terminal node size: 5
##        Average no. of terminal nodes: 28.273
## No. of variables tried at each split: 4
##               Total no. of variables: 12
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 169
##                             Analysis: RF-R
##                               Family: regr
##                       Splitting rule: mse *random*
##        Number of random split points: 10
##                 % variance explained: 99.29
##                           Error rate: 1404.75
```

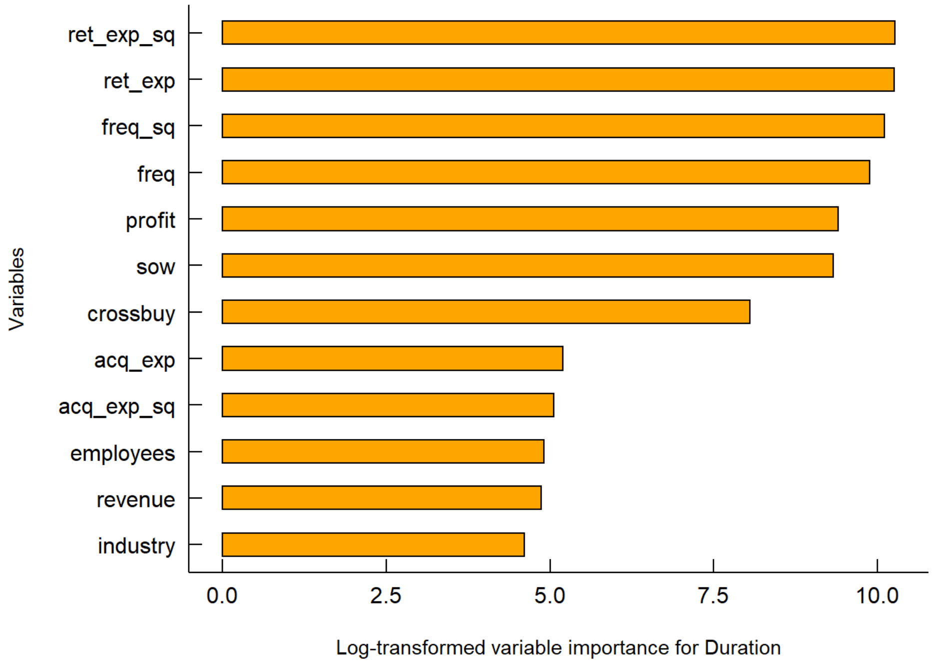
```
forest_duration$importance
```

```
##       profit      acq_exp   acq_exp_sq      ret_exp   ret_exp_sq         freq
## 1.202854e+04 8.132764e+01 5.792366e+01 2.829037e+04 2.864301e+04 1.949549e+04
##      freq_sq     crossbuy          sow     industry      revenue    employees
## 2.430503e+04 3.028284e+03 1.118423e+04 4.775466e-01 2.995912e+01 3.540989e+01
```

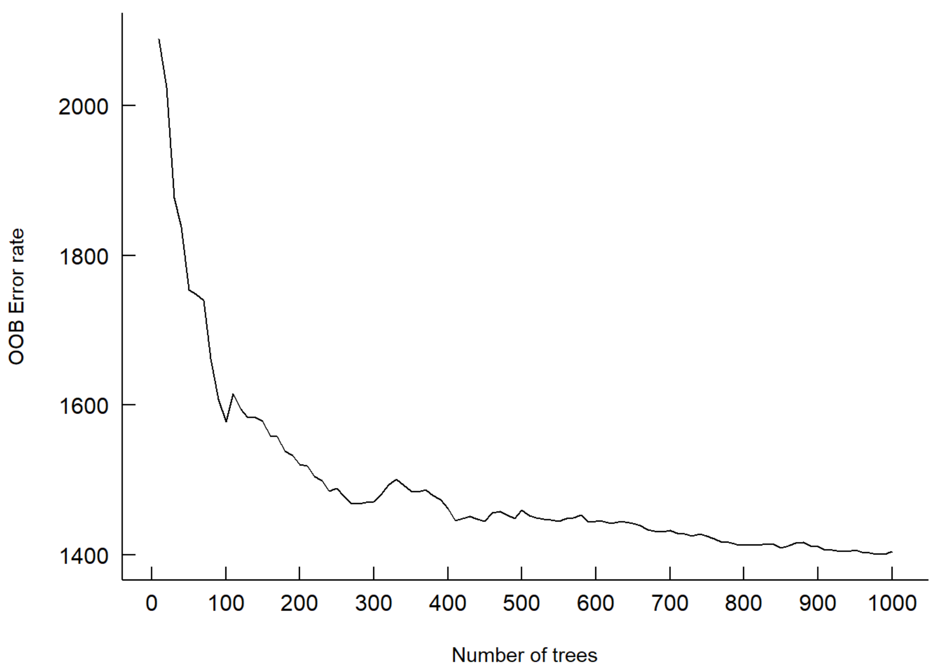
```
forest_duration$importance %>% log() # log transform
```

```
##     profit    acq_exp acq_exp_sq    ret_exp ret_exp_sq       freq    freq_sq
##  9.3950371  4.3984859  4.0591259 10.2502768 10.2626649  9.8779386 10.0984388
##   crossbuy        sow   industry    revenue  employees
##  8.0157515  9.3222596 -0.7390936  3.3998339  3.5669911
```

```
data.frame(importance = forest_duration$importance + 100) %>% # add a large +ve constant
  log() %>%
  tibble::rownames_to_column(var = "variable") %>%
  ggplot(aes(x = reorder(variable,importance), y = importance)) +
    geom_bar(stat = "identity", fill = "orange", color = "black", width = 0.5)+
    coord_flip() +
    labs(x = "Variables", y = "Log-transformed variable importance for Duration") +
    theme_nice
```




```
data.frame(err.rate = forest_duration$err.rate) %>%
  na.omit() %>%
  tibble::rownames_to_column(var = "trees") %>%
  mutate(trees = as.numeric(trees)) %>%
  ggplot(aes(x = trees, y = err.rate, group = 1))+
  geom_line()+
  scale_x_continuous(breaks = seq(0,1250,100))+
  labs(x = "Number of trees", y = "OOB Error rate")+
  theme_nice
```




## Optimized Selection

```r
# Establish a list of possible values for hyper-parameters
mtry.values <- seq(1,12,1)
nodesize.values <- seq(1,5,1)
ntree.values <- seq(1e3,6e3,1e3)

# Create a data frame containing all combinations
hyper_grid <- expand.grid(mtry = mtry.values, nodesize = nodesize.values, ntree = ntree.values)

# Create an empty vector to store OOB error values
oob_err <- c()

# Write a loop over the rows of hyper_grid to train the grid of modelsfor (i in 1:nrow(hyper_grid)) {
for (i in 1:nrow(hyper_grid)) {
    # Train a Random Forest model
   set.seed(123)
   model <- rfsrc(duration ~ profit + acq_exp + acq_exp_sq + ret_exp + ret_exp_sq + freq + freq_sq + crossbuy + sow +
industry + revenue +employees,
                            data = acq_train.df,
                            mtry = hyper_grid$mtry[i],
                            nodesize = hyper_grid$nodesize[i],
                            ntree = hyper_grid$ntree[i])


    # Store OOB error for the model
    oob_err[i] <- model$err.rate[length(model$err.rate)]
}

# Identify optimal set of hyperparmeters based on OOB error
opt_i <- which.min(oob_err)
print(hyper_grid[opt_i,])
```

```
##   mtry nodesize ntree
## 6    6        1  1000
```

```r
set.seed(100)

forest.hyper_duration <- rfsrc(duration ~ profit + acq_exp + acq_exp_sq + ret_exp + ret_exp_sq + freq + freq_sq + cro
ssbuy + sow + industry + revenue + employees,
                            data = acq_train.df,
                            mtry = 6,
                            nodesize = 1,
                            ntree = 1000,
                            importance = TRUE)
forest.hyper_duration
```

```
##                          Sample size: 268
##                      Number of trees: 1000
##            Forest terminal node size: 1
##        Average no. of terminal nodes: 140.294
## No. of variables tried at each split: 6
##               Total no. of variables: 12
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 169
##                             Analysis: RF-R
##                               Family: regr
##                       Splitting rule: mse *random*
##        Number of random split points: 10
##                 % variance explained: 99.47
##                           Error rate: 1033.72
```

# Interaction

```
find.interaction(forest.hyper_duration,
                      method = "vimp",
                      importance = "permute")
```

```
## Pairing ret_exp with ret_exp_sq
## Pairing ret_exp with freq
## Pairing ret_exp with freq_sq
## Pairing ret_exp with sow
## Pairing ret_exp with profit
## Pairing ret_exp with crossbuy
## Pairing ret_exp with acq_exp
## Pairing ret_exp with employees
## Pairing ret_exp with acq_exp_sq
## Pairing ret_exp with revenue
## Pairing ret_exp with industry
## Pairing ret_exp_sq with freq
## Pairing ret_exp_sq with freq_sq
## Pairing ret_exp_sq with sow
## Pairing ret_exp_sq with profit
## Pairing ret_exp_sq with crossbuy
## Pairing ret_exp_sq with acq_exp
## Pairing ret_exp_sq with employees
## Pairing ret_exp_sq with acq_exp_sq
## Pairing ret_exp_sq with revenue
## Pairing ret_exp_sq with industry
## Pairing freq with freq_sq
## Pairing freq with sow
## Pairing freq with profit
## Pairing freq with crossbuy
## Pairing freq with acq_exp
## Pairing freq with employees
## Pairing freq with acq_exp_sq
## Pairing freq with revenue
## Pairing freq with industry
## Pairing freq_sq with sow
## Pairing freq_sq with profit
## Pairing freq_sq with crossbuy
## Pairing freq_sq with acq_exp
## Pairing freq_sq with employees
## Pairing freq_sq with acq_exp_sq
## Pairing freq_sq with revenue
## Pairing freq_sq with industry
## Pairing sow with profit
## Pairing sow with crossbuy
## Pairing sow with acq_exp
## Pairing sow with employees
## Pairing sow with acq_exp_sq
## Pairing sow with revenue
## Pairing sow with industry
## Pairing profit with crossbuy
## Pairing profit with acq_exp
## Pairing profit with employees
## Pairing profit with acq_exp_sq
## Pairing profit with revenue
## Pairing profit with industry
## Pairing crossbuy with acq_exp
## Pairing crossbuy with employees
## Pairing crossbuy with acq_exp_sq
## Pairing crossbuy with revenue
## Pairing crossbuy with industry
## Pairing acq_exp with employees
## Pairing acq_exp with acq_exp_sq
## Pairing acq_exp with revenue
## Pairing acq_exp with industry
## Pairing employees with acq_exp_sq
## Pairing employees with revenue
## Pairing employees with industry
## Pairing acq_exp_sq with revenue
## Pairing acq_exp_sq with industry
## Pairing revenue with industry
##
```

```
##                          Method: vimp
##                  No. of variables: 12
##         Variables sorted by VIMP?: TRUE
##    No. of variables used for pairing: 12
##     Total no. of paired interactions: 66
##            Monte Carlo replications: 1
##     Type of noising up used for VIMP: permute
##
##                          Var 1      Var 2     Paired   Additive Difference
## ret_exp:ret_exp_sq    29162.0674 27781.6222 77061.7334 56943.6895 20118.0438
## ret_exp:freq          29162.0674 24575.8698 68111.5295 53737.9372 14373.5923
## ret_exp:freq_sq       29162.0674 21230.0776 61562.8800 50392.1449 11170.7350
## ret_exp:sow           29162.0674  9841.0873 44130.0836 39003.1547  5126.9290
## ret_exp:profit        29162.0674  7610.6515 44777.7263 36772.7189  8005.0075
## ret_exp:crossbuy      29162.0674  2439.9228 32714.6475 31601.9902  1112.6573
## ret_exp:acq_exp       29162.0674     8.7771 29265.8887 29170.8444    95.0442
## ret_exp:employees     29162.0674     7.0004 29008.1172 29169.0677  -160.9505
## ret_exp:acq_exp_sq    29162.0674    14.1276 29386.9569 29176.1950   210.7619
## ret_exp:revenue       29162.0674   -10.1128 29706.2562 29151.9546   554.3016
## ret_exp:industry      29162.0674    -0.3129 29377.5861 29161.7545   215.8317
## ret_exp_sq:freq       27689.5693 25140.5006 64703.7923 52830.0699 11873.7223
## ret_exp_sq:freq_sq    27689.5693 21331.3289 58757.0650 49020.8982  9736.1668
## ret_exp_sq:sow        27689.5693 10049.1674 43547.1938 37738.7367  5808.4571
## ret_exp_sq:profit     27689.5693  7587.2712 43251.3910 35276.8405  7974.5505
## ret_exp_sq:crossbuy   27689.5693  2289.6482 31549.0636 29979.2175  1569.8462
## ret_exp_sq:acq_exp    27689.5693     7.7812 28074.2247 27697.3505   376.8742
## ret_exp_sq:employees  27689.5693    24.7080 27577.2162 27714.2773  -137.0611
## ret_exp_sq:acq_exp_sq 27689.5693    10.1858 27051.9374 27699.7551  -647.8177
## ret_exp_sq:revenue    27689.5693   -25.2575 27628.5857 27664.3118   -35.7260
## ret_exp_sq:industry   27689.5693    13.0283 27532.6265 27702.5976  -169.9710
## freq:freq_sq          24329.7513 21244.3025 53648.7598 45574.0538  8074.7059
## freq:sow              24329.7513  9659.9582 36963.4081 33989.7095  2973.6986
## freq:profit           24329.7513  7475.8841 35696.1836 31805.6354  3890.5482
## freq:crossbuy         24329.7513  2385.8940 28016.3055 26715.6453  1300.6603
## freq:acq_exp          24329.7513    21.9528 24392.1532 24351.7041    40.4490
## freq:employees        24329.7513    17.3702 23649.8764 24347.1215  -697.2451
## freq:acq_exp_sq       24329.7513    11.7574 23659.1821 24341.5087  -682.3266
## freq:revenue          24329.7513   -11.2113 23886.3653 24318.5400  -432.1747
## freq:industry         24329.7513     3.2423 24469.8930 24332.9936   136.8994
## freq_sq:sow           20928.6061  9381.4070 33871.8585 30310.0131  3561.8454
## freq_sq:profit        20928.6061  7410.6120 33001.0761 28339.2181  4661.8580
## freq_sq:crossbuy      20928.6061  2222.7622 24322.7717 23151.3683  1171.4034
## freq_sq:acq_exp       20928.6061    20.9258 21075.5170 20949.5319   125.9851
## freq_sq:employees     20928.6061     9.4222 21214.3623 20938.0283   276.3340
## freq_sq:acq_exp_sq    20928.6061    15.5764 20675.1022 20944.1825  -269.0803
## freq_sq:revenue       20928.6061    -6.5836 20607.3730 20922.0225  -314.6495
## freq_sq:industry      20928.6061     7.8855 20151.4441 20936.4916  -785.0475
## sow:profit             9547.3327  7809.2866 19778.2804 17356.6193  2421.6611
## sow:crossbuy           9547.3327  2283.3808 12485.3739 11830.7135   654.6604
## sow:acq_exp            9547.3327     4.1886  9631.0344  9551.5213    79.5131
## sow:employees          9547.3327     6.7578  9664.0344  9554.0905   109.9439
## sow:acq_exp_sq         9547.3327    32.3528  9613.5027  9579.6855    33.8172
## sow:revenue            9547.3327     5.6753  9566.5262  9553.0080    13.5182
## sow:industry           9547.3327     9.8987  9330.2969  9557.2314  -226.9345
## profit:crossbuy        7731.5555  2370.0466 10382.3055 10101.6021   280.7034
## profit:acq_exp         7731.5555    15.6825  8011.9140  7747.2380   264.6760
## profit:employees       7731.5555     8.8034  7702.8616  7740.3589   -37.4973
## profit:acq_exp_sq      7731.5555     4.7379  8093.6608  7736.2934   357.3674
## profit:revenue         7731.5555   -17.2039  7823.2890  7714.3516   108.9374
## profit:industry        7731.5555     0.8602  7924.4565  7732.4157   192.0407
## crossbuy:acq_exp       2536.2765    20.0742  2643.6470  2556.3507    87.2963
## crossbuy:employees     2536.2765    19.7301  2451.3131  2556.0066  -104.6935
## crossbuy:acq_exp_sq    2536.2765    18.4907  2587.7264  2554.7672    32.9591
## crossbuy:revenue       2536.2765   -12.4676  2684.9829  2523.8089   161.1740
## crossbuy:industry      2536.2765     9.8129  2465.3070  2546.0894   -80.7824
## acq_exp:employees        19.3330    15.0031    36.5465    34.3361     2.2104
## acq_exp:acq_exp_sq       19.3330    38.0846    41.6116    57.4176   -15.8059
## acq_exp:revenue          19.3330   -14.2932   -13.7124     5.0398   -18.7521
```

```
## acq_exp:industry         19.3330    12.3581    52.6970    31.6911    21.0060
## employees:acq_exp_sq     11.0335     9.5527    29.5500    20.5862     8.9639
## employees:revenue        11.0335    -3.4563    24.1585     7.5772    16.5814
## employees:industry       11.0335    10.8080    18.9534    21.8415    -2.8881
## acq_exp_sq:revenue       41.8106   -10.2718    40.2954    31.5389     8.7565
## acq_exp_sq:industry      41.8106    11.5000    42.5628    53.3107   -10.7479
## revenue:industry         -3.7107    12.1593    24.5734     8.4485    16.1249
```

```
# Establish a list of possible values for hyper-parameters
mtry.values <- seq(1,12,1)
nodesize.values <- seq(1,5,1)
ntree.values <- seq(1e3,6e3,1e3)

# Create a data frame containing all combinations
hyper_grid <- expand.grid(mtry = mtry.values, nodesize = nodesize.values, ntree = ntree.values)

# Create an empty vector to store OOB error values
oob_err <- c()

# Write a loop over the rows of hyper_grid to train the grid of modelsfor (i in 1:nrow(hyper_grid)) {
for (i in 1:nrow(hyper_grid)) {
    # Train a Random Forest model
   set.seed(123)
   model <- rfsrc(duration ~ profit + acq_exp + acq_exp_sq + ret_exp + ret_exp_sq + freq + freq_sq + crossbuy + sow +
industry + revenue +employees + ret_exp*ret_exp_sq + ret_exp*freq + ret_exp*freq_sq + ret_exp_sq*freq + ret_exp_sq*fr
eq_sq + freq*freq_sq ,
                            data = acq_train.df,
                            mtry = hyper_grid$mtry[i],
                            nodesize = hyper_grid$nodesize[i],
                            ntree = hyper_grid$ntree[i])


    # Store OOB error for the model
    oob_err[i] <- model$err.rate[length(model$err.rate)]
}

# Identify optimal set of hyperparmeters based on OOB error
opt_i <- which.min(oob_err)
print(hyper_grid[opt_i,])
```

```
##   mtry nodesize ntree
## 6    6        1  1000
```

```
set.seed(100)

forest.hyper_duration2 <- rfsrc(duration  ~ profit + acq_exp + acq_exp_sq + ret_exp + ret_exp_sq + freq + freq_sq + c
rossbuy + sow + industry + revenue +employees + ret_exp*ret_exp_sq + ret_exp*freq + ret_exp*freq_sq + ret_exp_sq*freq
+ ret_exp_sq*freq_sq + freq*freq_sq,
                            data = acq_train.df,
                            mtry = 6,
                            nodesize = 1,
                            ntree = 1000,
                            importance = TRUE)
forest.hyper_duration2
```

```
##                          Sample size: 268
##                      Number of trees: 1000
##            Forest terminal node size: 1
##        Average no. of terminal nodes: 140.294
## No. of variables tried at each split: 6
##               Total no. of variables: 12
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 169
##                             Analysis: RF-R
##                               Family: regr
##                       Splitting rule: mse *random*
##        Number of random split points: 10
##                  % variance explained: 99.47
##                           Error rate: 1033.72
```

```
PredsDuration = predict.rfsrc(forest.hyper_duration,newdata = acq_test.df)$predicted
DurationDF <- data.frame(acq_test.df$duration, PredsDuration)
mse(acq_test.df$duration, PredsDuration)
```

```
## [1] 1525.854
```

```
MAE_D<-MAE(acq_test.df$duration, PredsDuration)
MAE_D
```

```
## [1] 25.36803
```

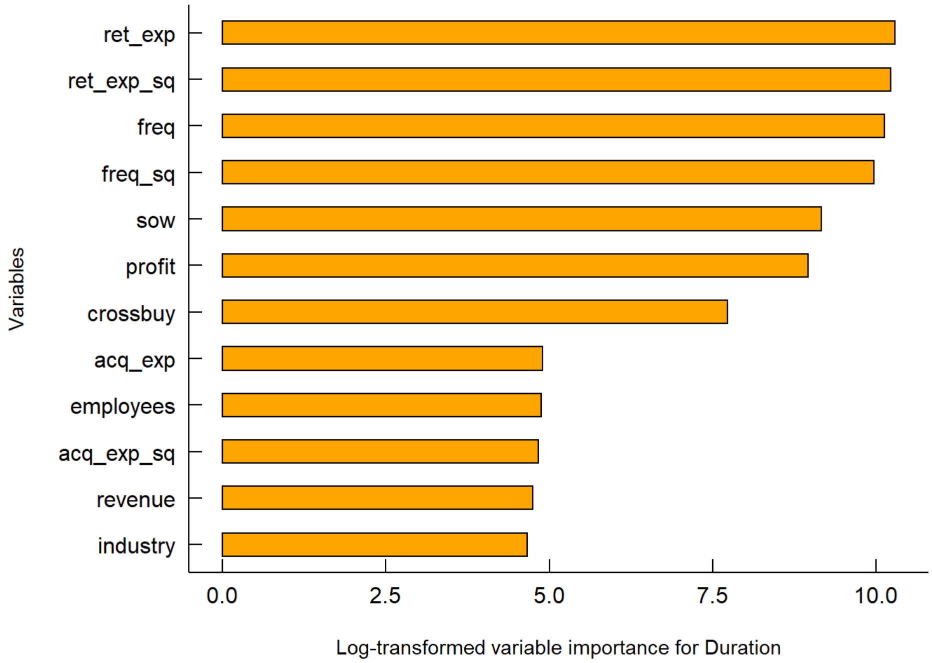
```
forest.hyper_duration$importance
```

```
##       profit      acq_exp   acq_exp_sq      ret_exp   ret_exp_sq         freq
##  7654.818215    33.448018    25.895889 29288.258733 27470.362939 24892.665145
##      freq_sq     crossbuy          sow     industry      revenue    employees
## 21207.728904  2166.546657  9480.508392     6.043773    15.641571    30.954803
```

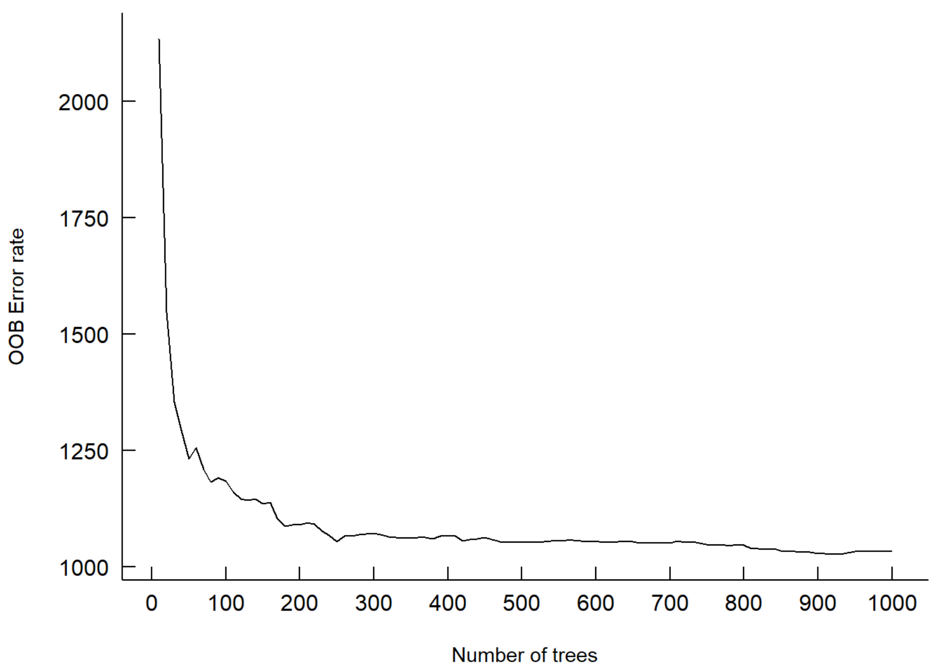
```
forest.hyper_duration$importance %>% log() # Log transform
```

```
##     profit    acq_exp acq_exp_sq    ret_exp ret_exp_sq       freq    freq_sq
##   8.943091   3.509993   3.254084  10.284942  10.220863  10.122328   9.962121
##   crossbuy        sow   industry    revenue  employees
##   7.680890   9.156993   1.799028   2.749932   3.432528
```

```
data.frame(importance = forest.hyper_duration$importance + 100) %>% # add a large +ve constant
  log() %>%
  tibble::rownames_to_column(var = "variable") %>%
  ggplot(aes(x = reorder(variable,importance), y = importance)) +
    geom_bar(stat = "identity", fill = "orange", color = "black", width = 0.5)+
    coord_flip() +
    labs(x = "Variables", y = "Log-transformed variable importance for Duration") +
    theme_nice
```




```
data.frame(err.rate = forest.hyper_duration$err.rate) %>%
  na.omit() %>%
  tibble::rownames_to_column(var = "trees") %>%
  mutate(trees = as.numeric(trees)) %>%
  ggplot(aes(x = trees, y = err.rate, group = 1))+
  geom_line()+
  scale_x_continuous(breaks = seq(0,1250,100))+
  labs(x = "Number of trees", y = "OOB Error rate")+
  theme_nice
```




# PDP Plots

# Duration: Retention Expenditure

```
min(forest.hyper_duration$xvar$ret_exp)
```

```
## [1] 0
```

```
max(forest.hyper_duration$xvar$ret_exp)
```

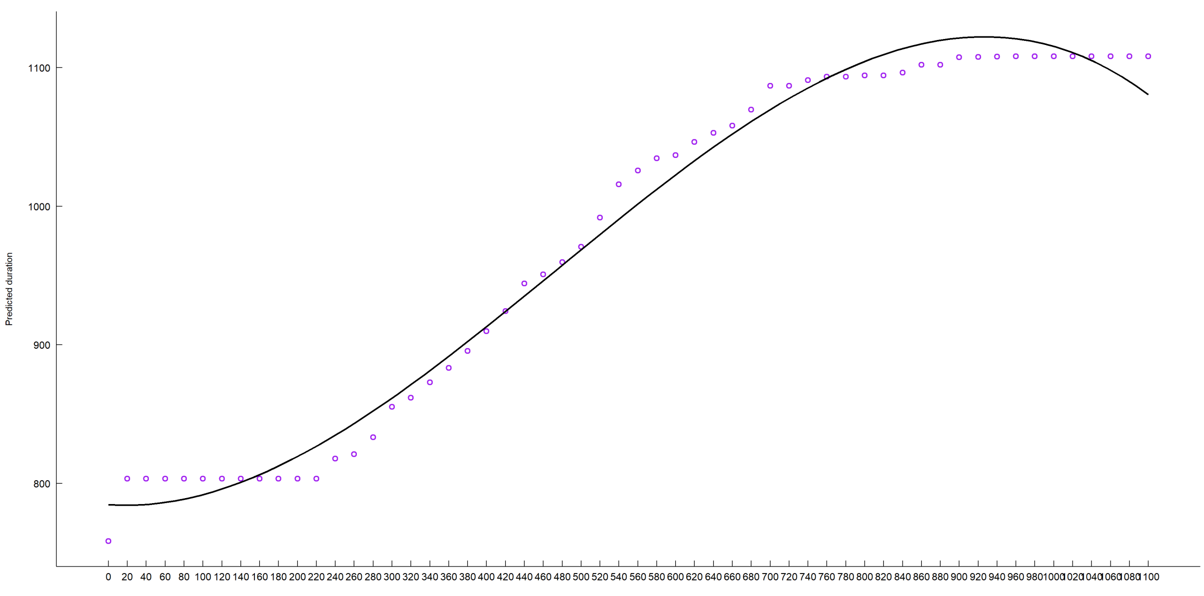
```
## [1] 1082.37
```

```
ret_exp_seq = seq(0,1100,20)
```

```
# extract marginal effect using partial dependence
marginal.effect <- partial(forest.hyper_duration,
                           partial.xvar = "ret_exp",
                           partial.values = ret_exp_seq)

means.exp <- marginal.effect$regrOutput$duration %>% colMeans()
```

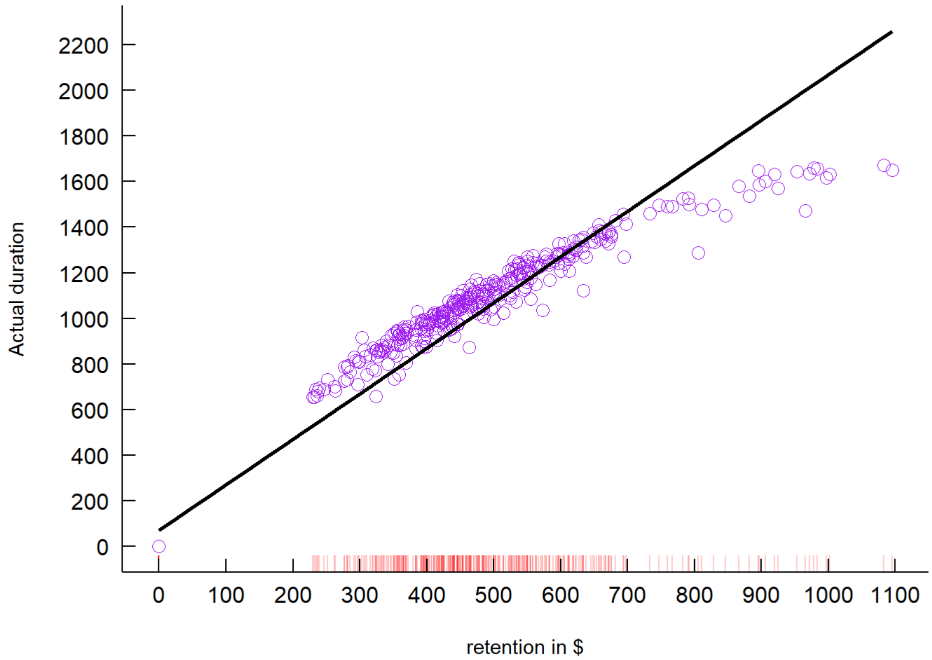
```
marginal.effect.df <-
  data.frame(pred.duration = means.exp, ret_exp_seq = ret_exp_seq)
```

```
ggplot(marginal.effect.df, aes(x = ret_exp_seq, y = pred.duration)) +
  geom_point(shape = 21, color = "purple", size = 2, stroke = 1.2)+
  geom_smooth(method = "lm", formula = y ~ poly(x,3), se = FALSE, color = "black")+ # try with other values
  labs(x = "Average retention in $", y = "Predicted duration") +
  scale_x_continuous(breaks = seq(0,1100,20))+
  theme_nice # positive effect of ret_exp not clear as suggested by reg coefs
```

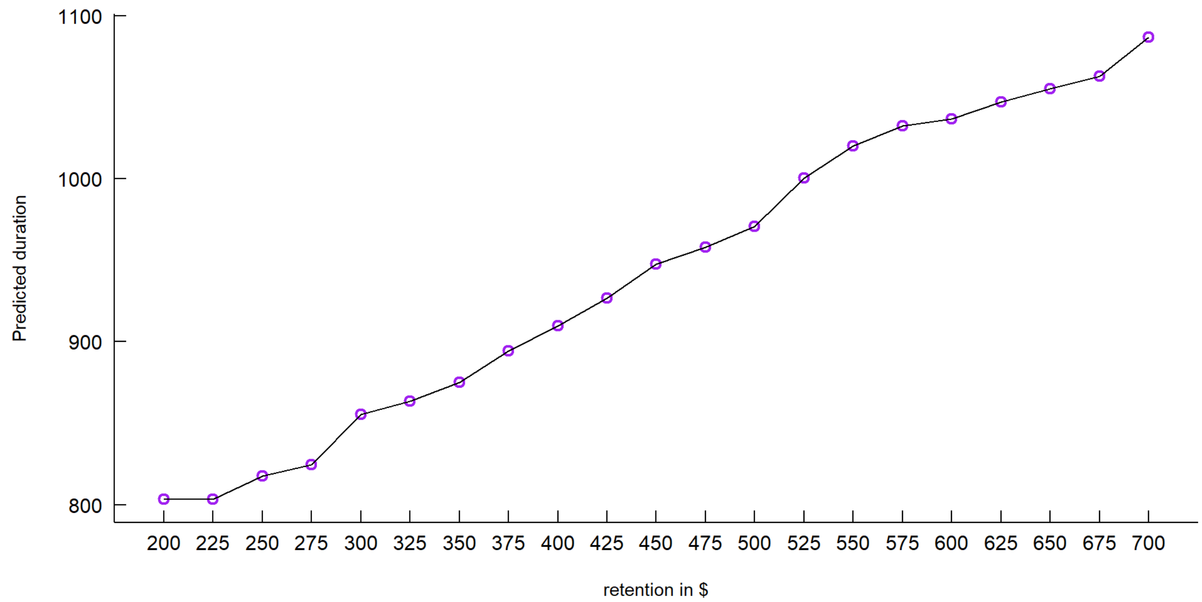


```
# first check relationship between actual duration and ret_exp

ggplot(acquisitionRetention, aes(x = ret_exp, y = duration)) +
  geom_point(shape = 21, col = "purple", size = 3) +
  stat_smooth(method = "lm", se = FALSE, color = "black") +
  scale_x_continuous(breaks = seq(0,1100,100)) +
  scale_y_continuous(breaks = seq(0,2200,200)) +
  geom_rug(sides = "b", col = "red", alpha = 0.2) +
  labs(y = "Actual duration", x = "retention in $") +
  theme_nice
```

```
## `geom_smooth()` using formula 'y ~ x'
```

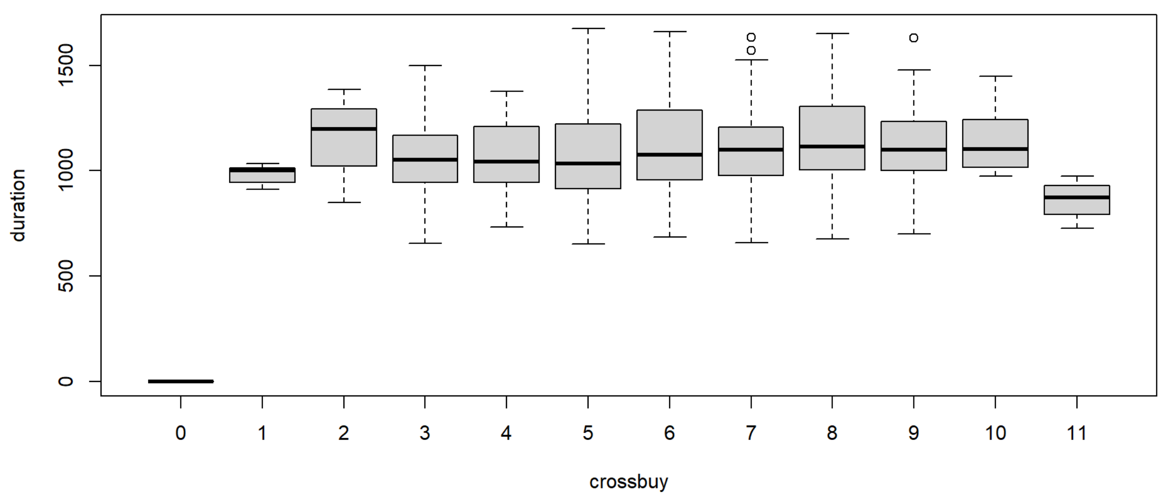



```
# repeat with smaller values of ret_exp
ret_exp_seq2 = seq(200,700,25)

marginal.effect.new <- partial(forest.hyper_duration,
                           partial.xvar = "ret_exp",
                           partial.values = ret_exp_seq2)

means.exp.new <- marginal.effect.new$regrOutput$duration %>% colMeans()

marginal.effect.df.new <-
  data.frame(pred.duration = means.exp.new, ret_exp_seq = ret_exp_seq2)

ggplot(marginal.effect.df.new, aes(x = ret_exp_seq, y = pred.duration)) +
  geom_point(shape = 21, color = "purple", size = 2, stroke = 1.2)+
  geom_path()+
  labs(x = "retention in $", y = "Predicted duration") +
  scale_x_continuous(breaks = seq(200,700,25))+
  theme_nice
```




# Duration: Crossbuy Categories

```
CrossbuyPlot <- boxplot(data=acquisitionRetention, duration ~ crossbuy)
```




```
x <- c("Minimum", "25th Percentile", "Median", "75th Percentile", "Maximum")

CrossbuyStats <- data.frame(x, CrossbuyPlot$stats)
colnames(CrossbuyStats) <- c("Statistic","0","1","2","3","4","5","6","7", "8", "9", "10", "11")
```

# Duration: Retention Expenditure Squared

```
min(forest.hyper_duration$xvar$ret_exp_sq)
```

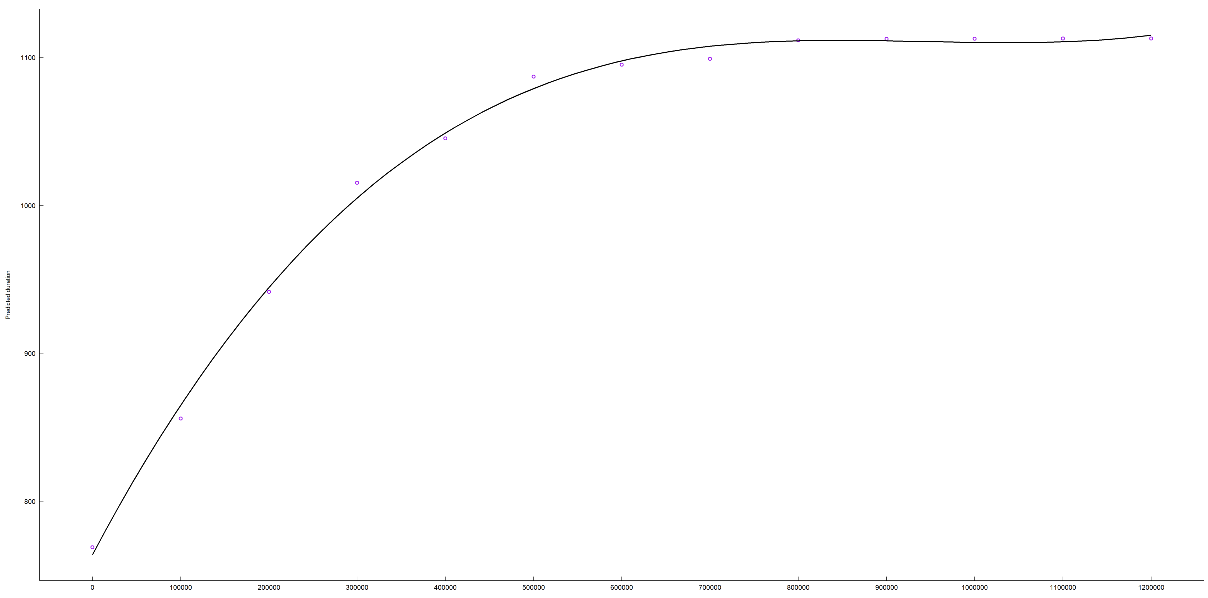
```
## [1] 0
```

```
max(forest.hyper_duration$xvar$ret_exp_sq)
```

```
## [1] 1171525
```

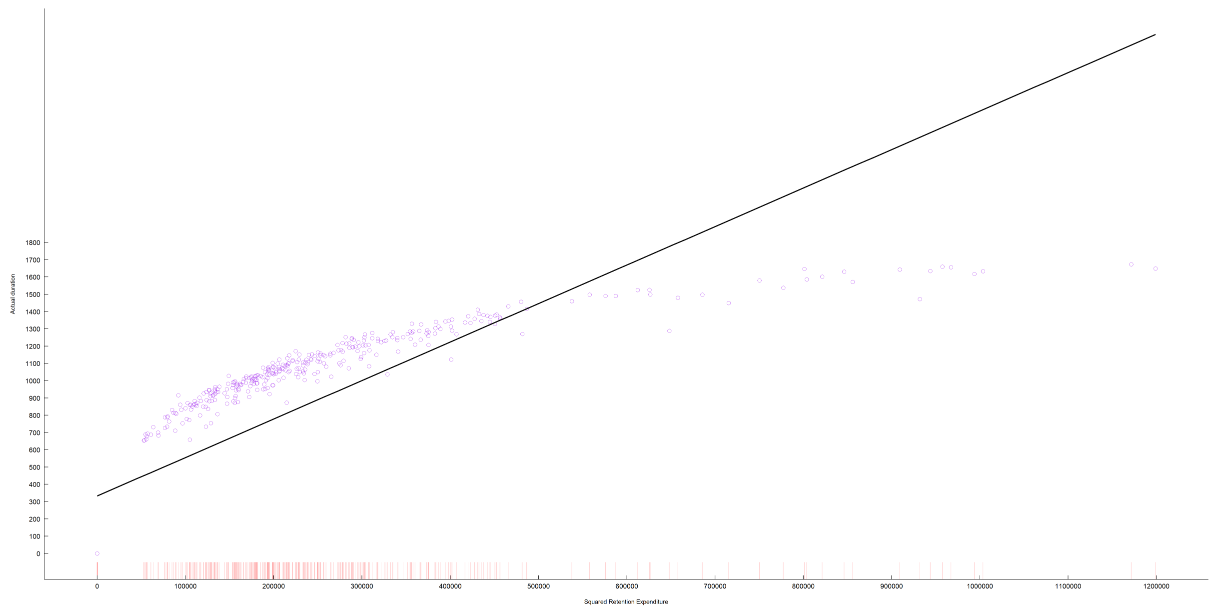
```
ret_sq_seq = seq(0,1200000,100000)

# extract marginal effect using partial dependence
marginal.effect <- partial(forest.hyper_duration,
                           partial.xvar = "ret_exp_sq",
                           partial.values = ret_sq_seq)

means.exp <- marginal.effect$regrOutput$duration %>% colMeans()

marginal.effect.df <-
  data.frame(pred.duration = means.exp, ret_sq_seq = ret_sq_seq)
```

```
ggplot(marginal.effect.df, aes(x = ret_sq_seq, y = pred.duration)) +
  geom_point(shape = 21, color = "purple", size = 2, stroke = 1.2)+
  geom_smooth(method = "lm", formula = y ~ poly(x,3), se = FALSE, color = "black")+ # try with other values
  labs(x = "Squared Retention Expenditure", y = "Predicted duration") +
  scale_x_continuous(breaks = seq(0,1200000,100000))+
  theme_nice # positive effect of ret_exp not clear as suggested by reg coefs
```

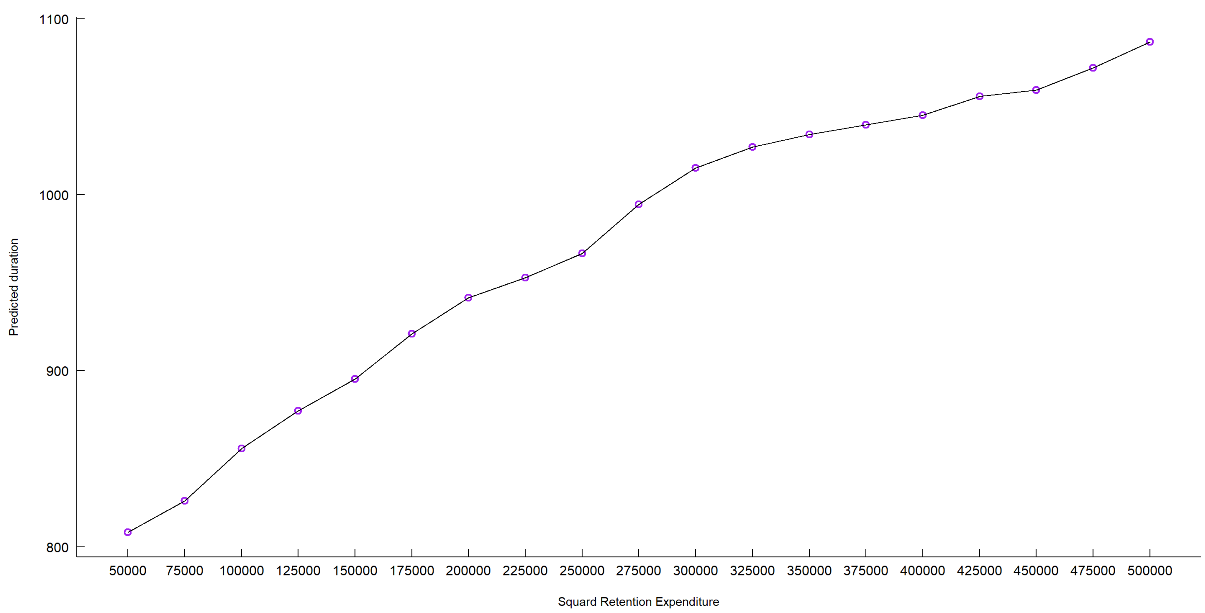



```
# first check relationship between actual duration and ret_exp

ggplot(acquisitionRetention, aes(x = ret_exp_sq, y = duration)) +
  geom_point(shape = 21, col = "purple", size = 3) +
  stat_smooth(method = "lm", se = FALSE, color = "black") +
  scale_x_continuous(breaks = seq(0,1200000,100000)) +
  scale_y_continuous(breaks = seq(0,1800,100)) +
  geom_rug(sides = "b", col = "red", alpha = 0.2) +
  labs(y = "Actual duration", x = "Squared Retention Expenditure") +
  theme_nice
```

```
## `geom_smooth()` using formula 'y ~ x'
```

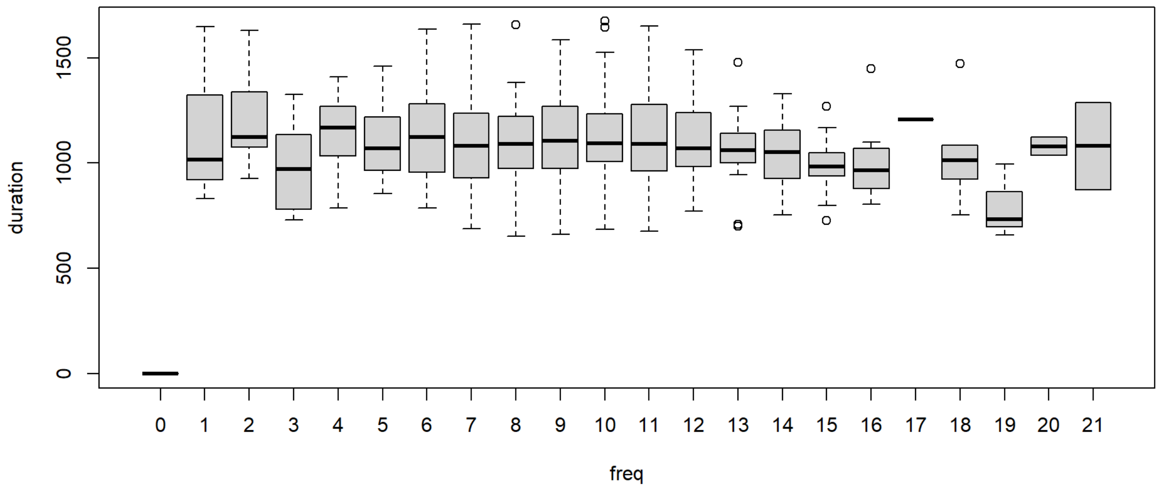



```
# repeat with smaller values of ret_exp
ret_sq_seq2 = seq(50000,500000,25000)

marginal.effect.new <- partial(forest.hyper_duration,
                          partial.xvar = "ret_exp_sq",
                          partial.values = ret_sq_seq2)

means.exp.new <- marginal.effect.new$regrOutput$duration %>% colMeans()

marginal.effect.df.new <-
  data.frame(pred.duration = means.exp.new, ret_sq_seq = ret_sq_seq2)

ggplot(marginal.effect.df.new, aes(x = ret_sq_seq, y = pred.duration)) +
  geom_point(shape = 21, color = "purple", size = 2, stroke = 1.2)+
  geom_path()+
  labs(x = "Squard Retention Expenditure", y = "Predicted duration") +
  scale_x_continuous(breaks = seq(50000,500000,25000))+
  theme_nice
```



# Duration: frequency

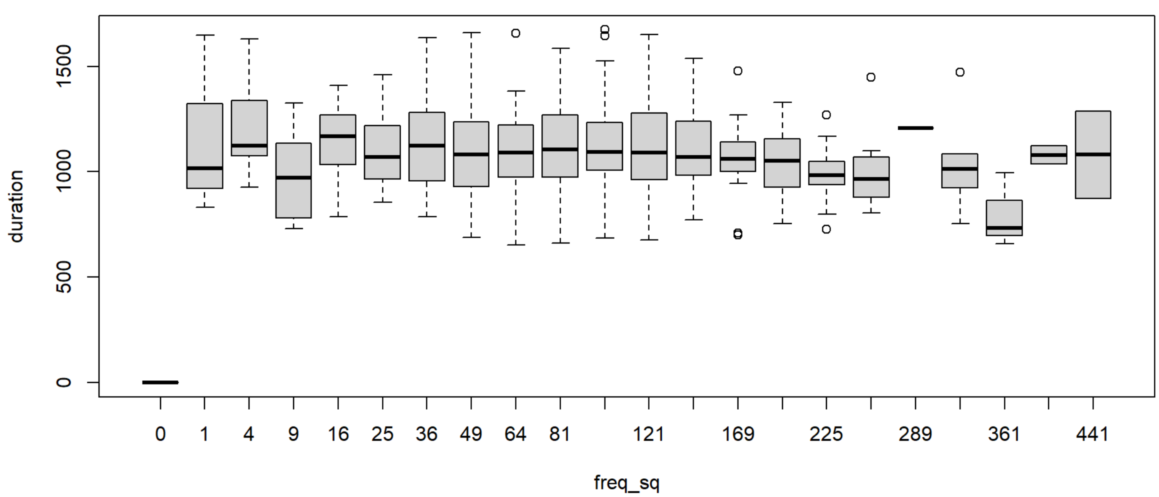
```
FreqPlot <- boxplot(data=acquisitionRetention, duration ~ freq)
```




```
x <- c("Minimum", "25th Percentile", "Median", "75th Percentile", "Maximum")

FreqStats <- data.frame(x, FreqPlot$stats)
colnames(FreqStats) <- c("Statistic","0","1","2","3","4","5","6","7", "8", "9", "10", "11", "12", "13", "14", "15",
"16", "17", "18", "19", "20", "21")
```

# Duration: frequency squared

```
FreqSqPlot <- boxplot(data=acquisitionRetention, duration ~ freq_sq)
```



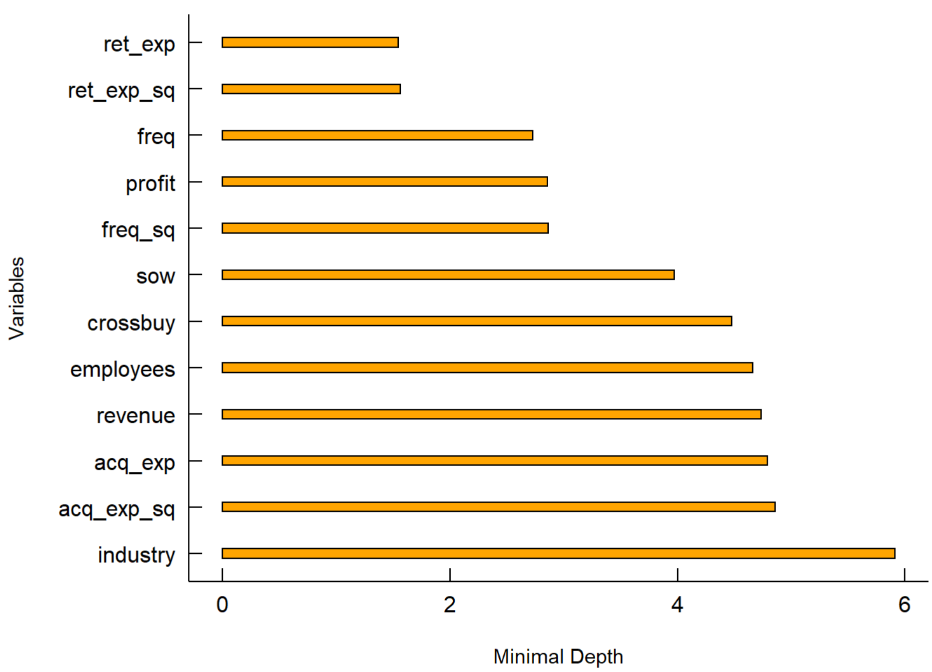
```
x <- c("Minimum", "25th Percentile", "Median", "75th Percentile", "Maximum")

FreqSqStats <- data.frame(x, FreqSqPlot$stats)
colnames(FreqSqStats) <- c("Statistic","0","1","4","9","16","25","36","49", "64", "81", "100", "121", "144", "169",
"196", "225", "256", "289", "324", "361", "400", "441")
```

# Minimal Depth

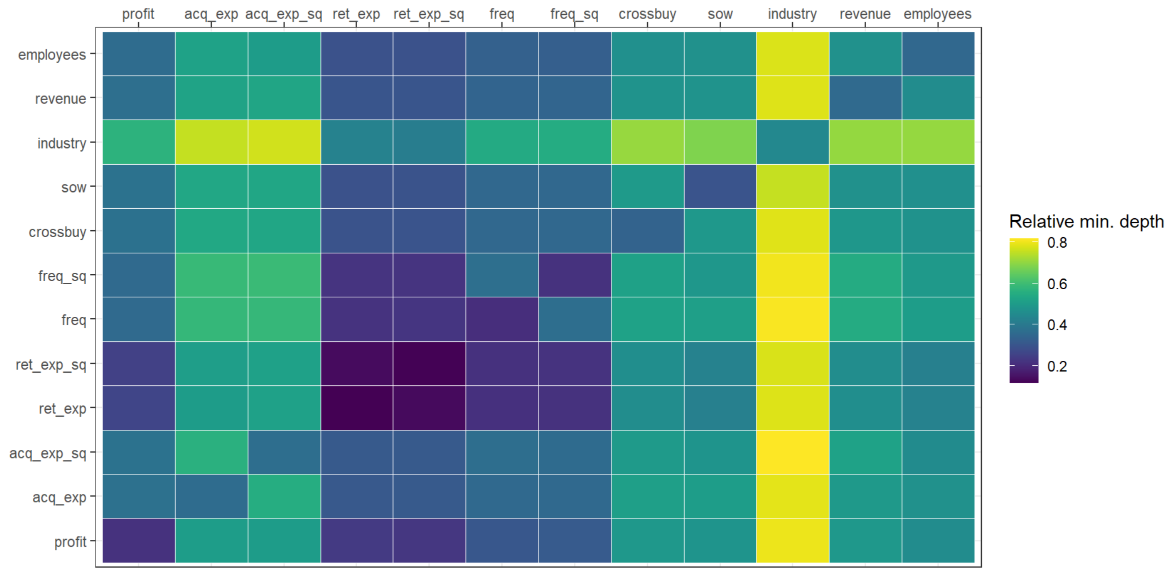
```
mindepth <- max.subtree(forest.hyper_duration, sub.order = TRUE)

print(round(mindepth$order, 3)[,1])
```

```
##     profit    acq_exp acq_exp_sq    ret_exp ret_exp_sq       freq    freq_sq
##      2.854      4.793      4.860      1.544      1.562      2.728      2.865
##   crossbuy        sow   industry    revenue  employees
##      4.478      3.974      5.911      4.736      4.659
```

```
data.frame(md = round(mindepth$order, 3)[,1]) %>%
  tibble::rownames_to_column(var = "variable") %>%
  ggplot(aes(x = reorder(variable,desc(md)), y = md)) +
    geom_bar(stat = "identity", fill = "orange", color = "black", width = 0.2)+
    coord_flip() +
     labs(x = "Variables", y = "Minimal Depth")+
     theme_nice
```




```
as.matrix(mindepth$sub.order) %>%
  reshape2::melt() %>%
  data.frame() %>%
  ggplot(aes(x = Var1, y = Var2, fill = value)) +
    scale_x_discrete(position = "top") +
    geom_tile(color = "white") +
    viridis::scale_fill_viridis("Relative min. depth") +
    labs(x = "", y = "") +
    theme_bw()
```


profit
acq_exp
acq_exp_sq
ret_exp
ret_exp_sq
freq
freq_sq
crossbuy
sow
industry
revenue
employees
Relative min. depth
0.8
0.6
0.4
0.2