Predicting Employee Attrition for Werner Enterprises In Advance

Ali Al-Ghaithi, Mackenzie Maschka, Mamadou Sagnon, & Mamadou Traore

ECON 8310 Business Forecasting

Fall 2020

What’s the problem?

Werner Enterprises, a transportation and logistics company, has **difficulty retaining freight drivers** (as is the trend throughout the industry). Sometimes Werner has very **little notice** that a drive will quit before it happens. This can **interrupt Werner’s workflow** and **can cost the company time and money**.

How can Werner know of these quitting drivers early enough to either persuade the driver to stay or find a replacement?

Our Goal

The goal of this project is to:

- Create a model to accurately predict which drivers are going to quit 30 days in advance

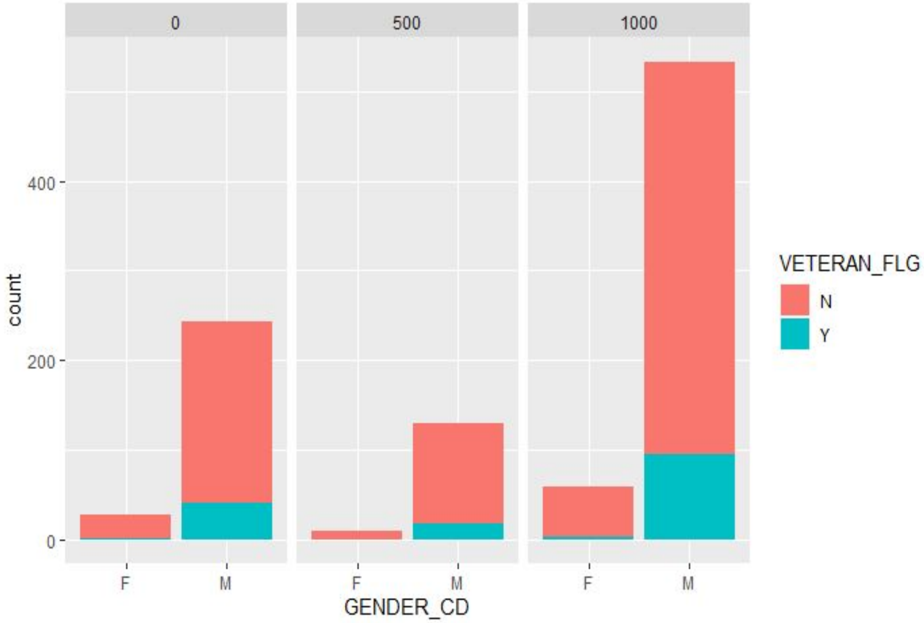
Let’s learn about the driver

	Unique_ID <int>	Age <int>	GENDER_CD <fctr>	VETERAN_FLG <fctr>	Max_Students <fctr>	STATE_CD <fctr>	REHIRED <fctr>	Retention..Days. <int>
1	100277	53	M	N	1000	VA	N	1421
2	100370	26	M	N	0	NV	Y	351
3	100455	26	M	N	0	FL	Y	727
4	100536	45	M	Y	0	GA	Y	367
5	100553	43	M	N	1000	FL	N	557
6	100839	28	M	N	0	OH	N	242

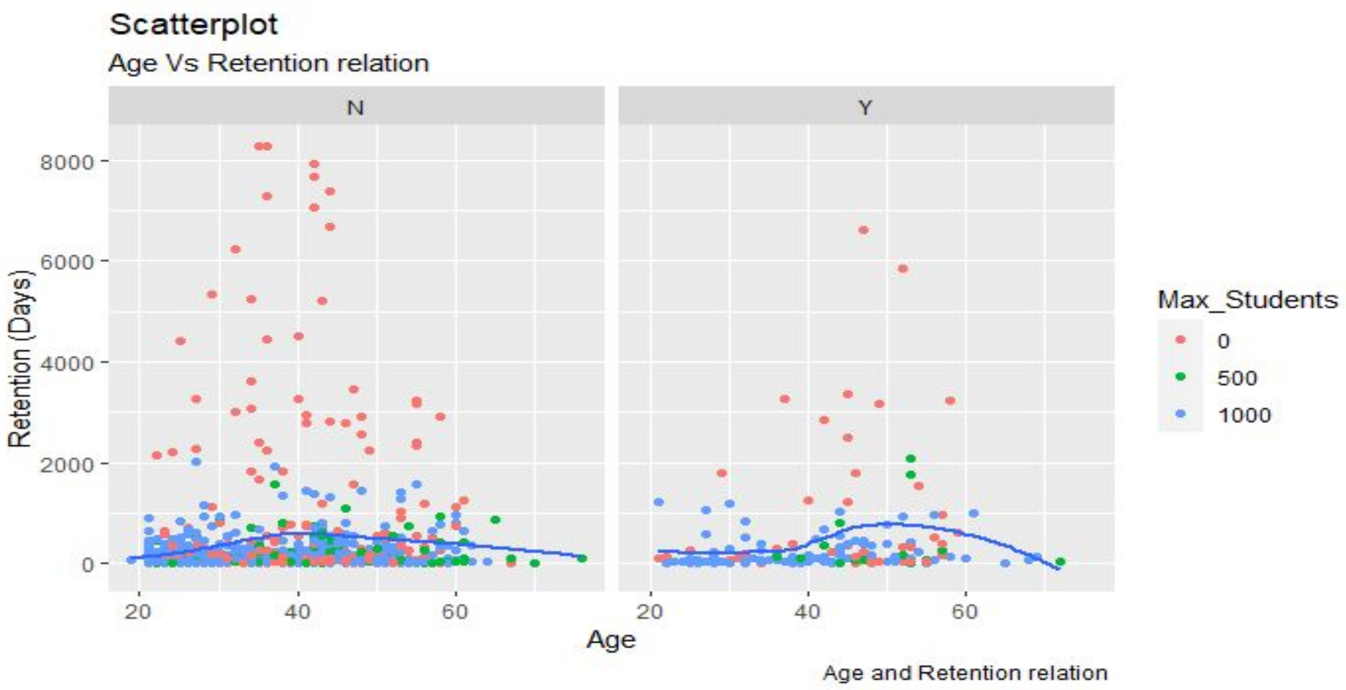
Bar plots for student, veteran and gender

Proportion of female and male

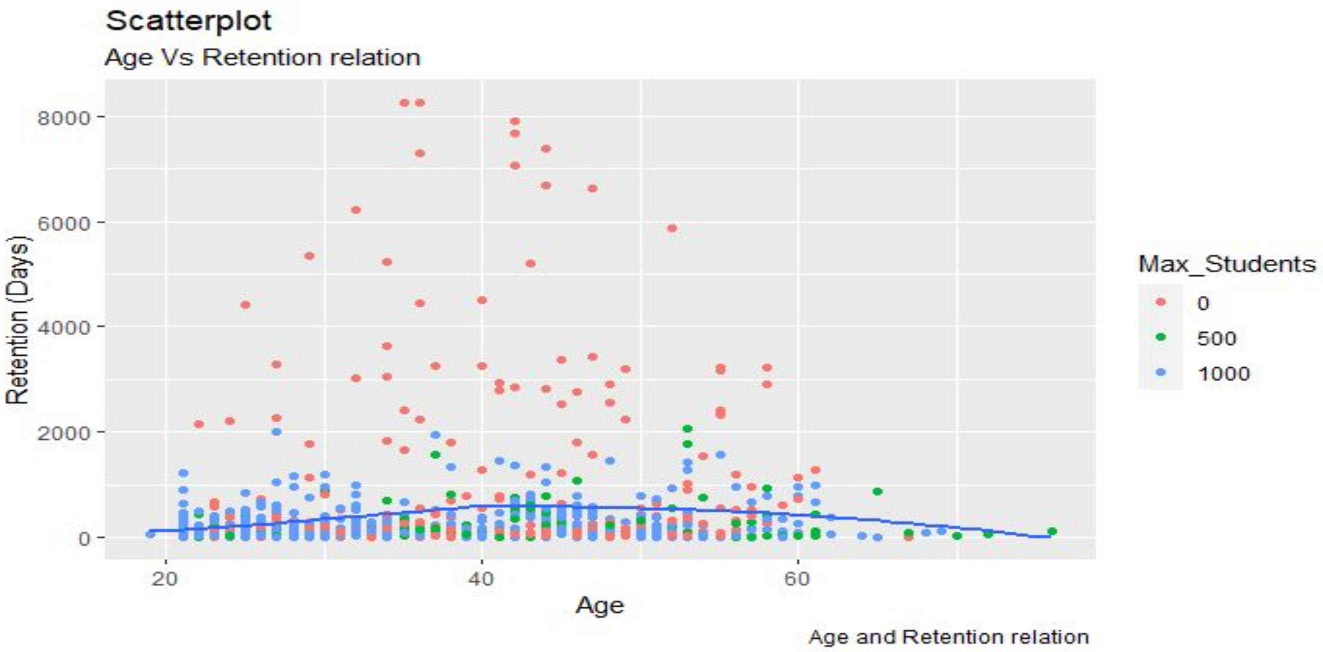
For max student and veteran.



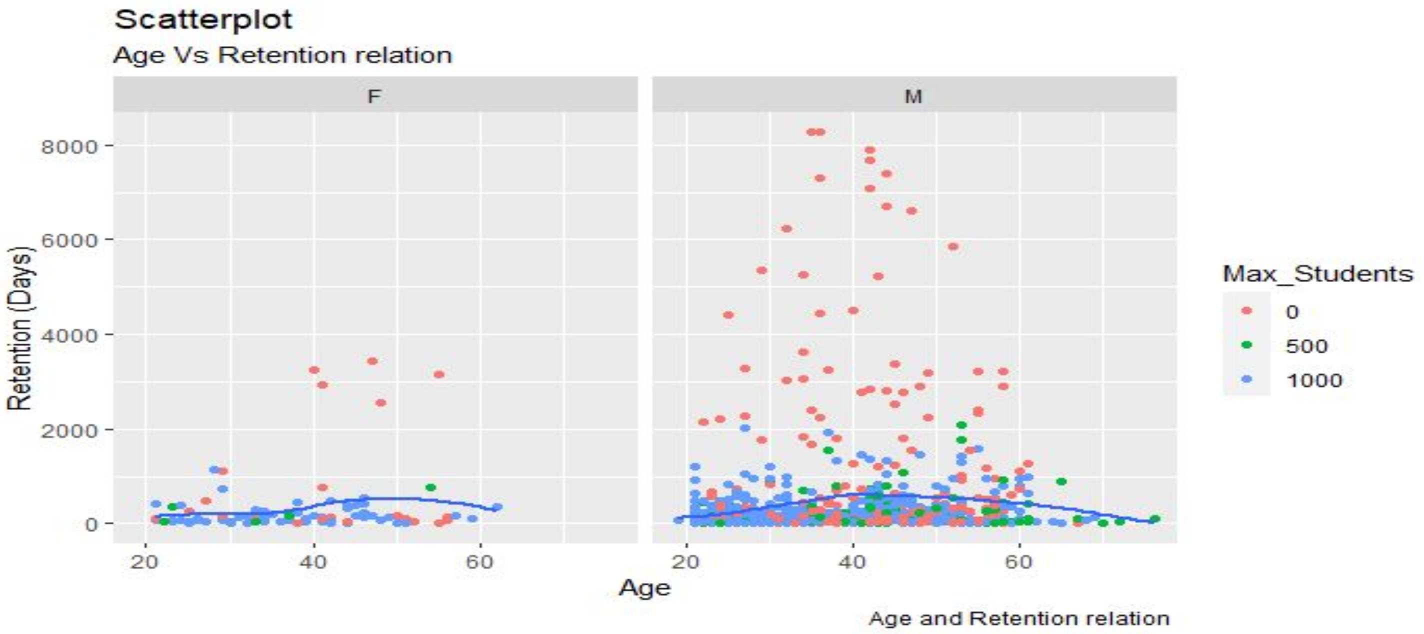
Scatterplot for student, Retention, veteran and Age



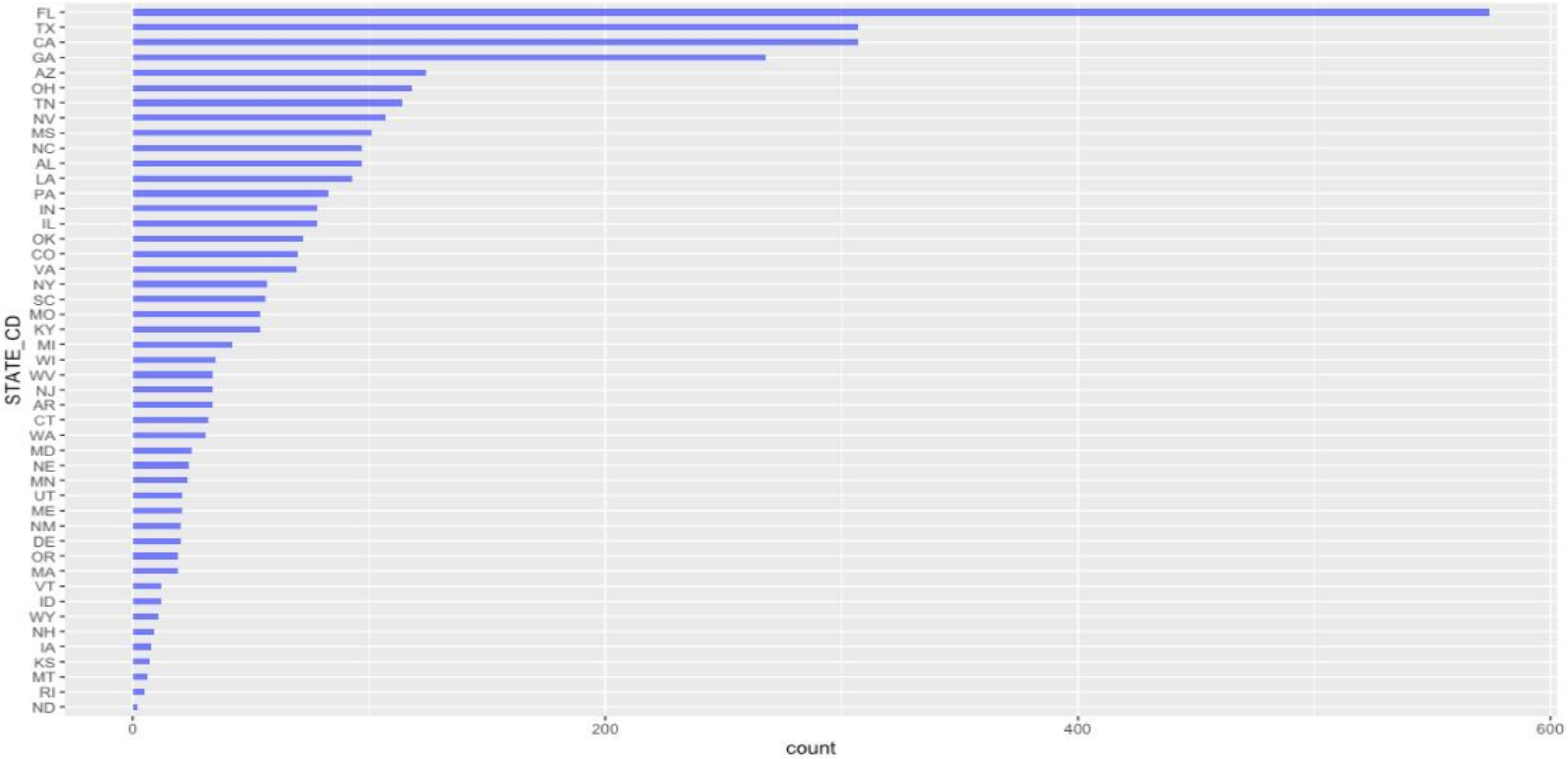
Relation between age, retention, max_student



Relation between Age, Retention, student and gender



Our data Drivers



Defining the Dependent Variable

To define a dependent variable for employees who will quit within 30 days, we:

1. Created a variable called ‘daysLeft’

   daysLeft = FiredDate - ReportDate

2. Flagged when this ‘daysLeft’ was less than 30 with a variable called ‘quitIn30Days’

$$\textbf{quitIn30Days} = \begin{cases} 0 \text{ if the employee } \textbf{will not} \text{ quit in 30 days} \\ 1 \text{ if the employee } \textbf{will} \text{ quit in 30 days} \end{cases}$$

Data Splitting

Size of the data

```
[1] 69340    81
```

		0	1
Whole Data		59057	10283
Train Data=70%		41340	7199
Test Data=30%		17717	3084

Variables We Used

```
 [1] "Age"                          "GENDER_CD"                    "Zip5"
 [4] "EQUIPMENT_COST_DIVISION_CD"  "percent_miles_pay"            "FiredToday"
 [7] "Vet"                          "daysWorked"                   "quitIn30Days"
[10] "hireType_500"                 "hireType_1000"                "state_AR"
[13] "state_AZ"                     "state_CA"                     "state_CO"
[16] "state_CT"                     "state_DE"                     "state_FL"
[19] "state_GA"                     "state_IA"                     "state_ID"
[22] "state_IL"                     "state_IN"                     "state_KS"
[25] "state_KY"                     "state_LA"                     "state_MA"
[28] "state_MD"                     "state_ME"                     "state_MI"
[31] "state_MN"                     "state_MO"                     "state_MS"
[34] "state_MT"                     "state_NC"                     "state_ND"
[37] "state_NE"                     "state_NH"                     "state_NJ"
[40] "state_NM"                     "state_NV"                     "state_NY"
[43] "state_OH"                     "state_OK"                     "state_OR"
[46] "state_PA"                     "state_RI"                     "state_SC"
[49] "state_TN"                     "state_TX"                     "state_UT"
[52] "state_VA"                     "state_VT"                     "state_WA"
[55] "state_WI"                     "state_WV"                     "state_WY"
[58] "driver_Regional"              "driver_Specialized.Regional" "driver_Team"
[61] "driver_Trainee"               "day_1"                        "day_2"
[64] "day_3"                        "day_4"                        "day_5"
[67] "day_6"                        "month_2"                      "month_3"
[70] "month_4"                      "month_5"                      "month_6"
[73] "month_7"                      "month_8"                      "month_9"
[76] "month_10"                     "month_11"                     "month_12"
[79] "year_2015"                    "year_2016"                    "year_2017"
```

The Issue of Unbalanced Data

Whole Data:

0	1
59057	10283

Majority class almost 6x larger than minority class!

- Accuracy vs weighted accuracy

Measurements we used for the unbalanced data:

- Precision
- Recall (sensitivity)
- F
- G-means
- Weighted Accuracy

Higher is better!

Imputing Missing Values

- Zip5: only two drivers have missing zip codes

  Imputed based on the average of zip codes of drivers from the same state

- EQUIPMENT_COST_DIVISION_CD:

  Imputed by the most common one

- percent_miles_pay:

  Next Obs. Carried Backward method

Model Time!

Logistic Regression

```
## model
model = sm.Logit(y, x)

## fitted model
modelFit = model.fit()

## prediction
pred = modelFit.predict(xt)
pred[pred > 0.5] = 1
pred[pred <= 0.5] = 0
```

Confusion Matrix for Training Set:

	Pred 0	Pred 1
Actual 0	41109	231
Actual 1	6557	642

```
array([[41109,   231],
       [ 6557,   642]])
```

Confusion Matrix for Validation Set:

	Pred 0	Pred 1
Actual 0	17625	92
Actual 1	2817	267

```
array([[17625,    92],
       [ 2817,   267]])
```

Training:

Method	Acc_Positive(Recall)	Acc_Negative	Precision	F_measure	G_mean	Weighted_Accuracy
logit	0.089179	0.994412	0.735395	0.159068	0.297793	0.541796

Validation:

Method	Acc_Positive(Recall)	Acc_Negative	Precision	F_measure	G_mean	Weighted_Accuracy
logit	0.086576	0.994807	0.743733	0.155097	0.293473	0.540692

Boosting

```
# Generate the boosting model
model = GradientBoostingClassifier(n_estimators=100, max_depth=10, min_samples_leaf=10, random_state=42)
```

Confusion Matrix for Training Set:

	Pred 0	Pred 1
Actual 0	41180	160
Actual 1	1181	6018

Confusion Matrix for Validation Set:

	Pred 0	Pred 1
Actual 0	17558	159
Actual 1	794	2290

Training:

Method	Acc_Positive(Recall)	Acc_Negative	Precision	F_measure	G_mean	Weighted_Accuracy
boosting	0.835949	0.99613	0.974102	0.899753	0.912532	0.91604

Validation:

Method	Acc_Positive(Recall)	Acc_Negative	Precision	F_measure	G_mean	Weighted_Accuracy
boosting	0.742542	0.991026	0.935076	0.827761	0.857833	0.866784

kNN

```python
## model
model = KNeighborsClassifier(n_neighbors=10, n_jobs = -1, metric='euclidean')
```

Confusion Matrix for Training Set:

	Pred 0	Pred 1	
array([[41132,		208],	Actual 0
[	803,	6396]])	Actual 1

Confusion Matrix for Validation Set:

	Pred 0	Pred 1	
array([[17554,		163],	Actual 0
[	513,	2571]])	Actual 1

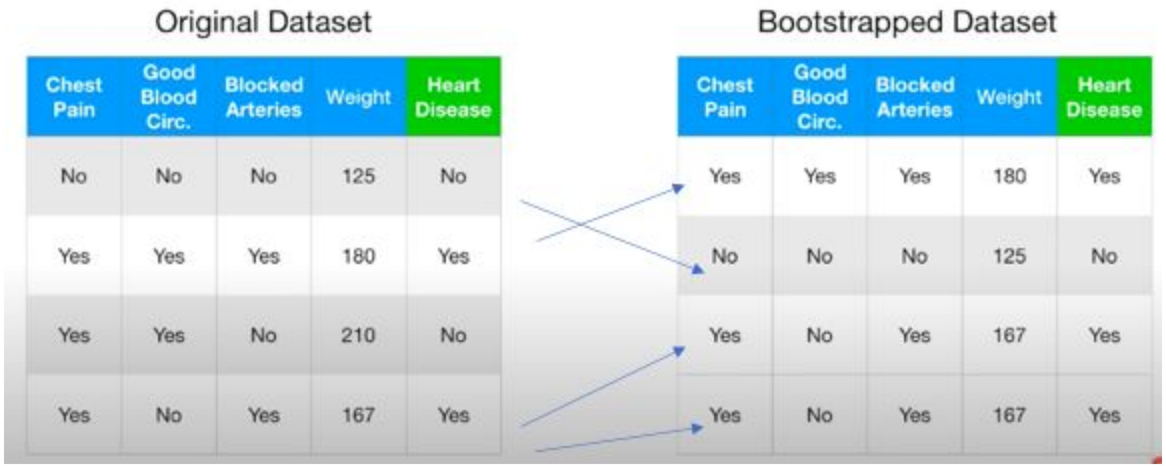
Training:

Method	Acc_Positive(Recall)	Acc_Negative	Precision	F_measure	G_mean	Weighted_Accuracy
kNN	0.888457	0.994969	0.968504	0.926755	0.940206	0.941713

Validation:

Method	Acc_Positive(Recall)	Acc_Negative	Precision	F_measure	G_mean	Weighted_Accuracy
kNN	0.833658	0.9908	0.94038	0.883809	0.908839	0.912229

Random Forest without tuning

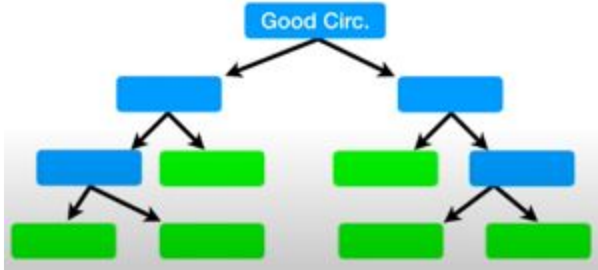


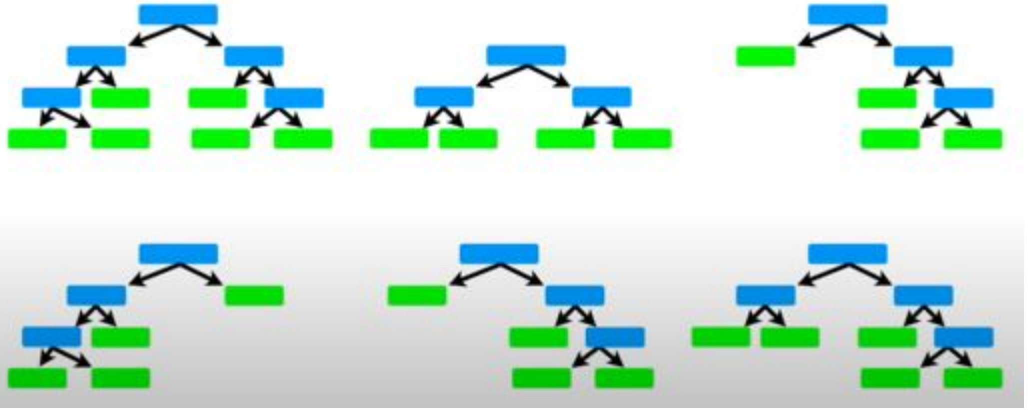
Original Dataset

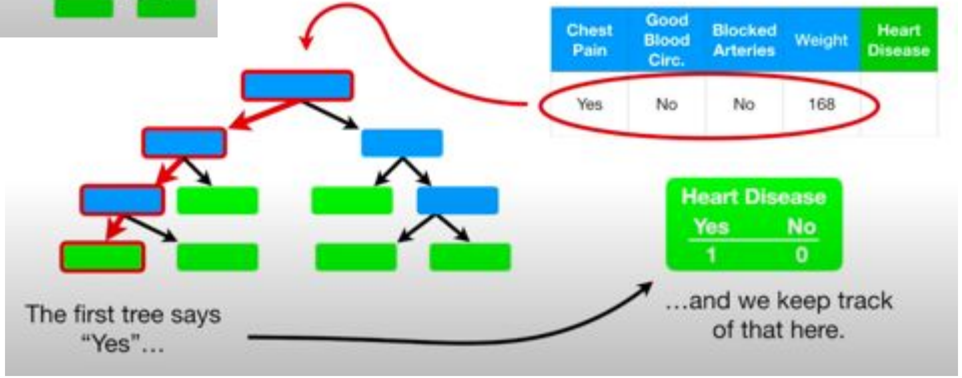
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes






Chest Pain
Good Blood Circ.
Blocked Arteries
Weight
Heart Disease
Yes
No
No
168
Heart Disease
Yes No
1 0
The first tree says "Yes"...
...and we keep track of that here.

Tuning Random Forest

1. Cut-off:

•Each tree gives a classification for each observation, which counts as a vote for the classification.

• By aggregating the votes from all the trees, RF decides the winning class as the one having the most votes for the observation.

• The cut-off, which is 0.5 by default when there are two classes, is used to adjust the votes.

•**For example:** suppose an observation has 209 votes for the first class and only 92 votes for the second class. If the default cut-off is used, the winning class is decided by comparing 209/0.5 = 408 to 92/0.5 = 184. Since 408 is greater than 194, the winning class is the first class. If a higher cut-off is used, say 0.7, then 209/0.7 = 298.6 is compared to 92/(1 − 0.7) = 306.67 and the winning class is the second class.

2. Sample size:

•In the Random Forest, when each tree is fitted to a bootstrap sample of the original training dataset, a class imbalance leads to there being only a small number of the minority class in the bootstrap sample, which results in poor predicting performance for the minority class (poor sensitivity). To alleviate this problem, the number of samples drawn from each class in RF can be tuned so that they are equal, which forces the classes to be balanced

•The numbers of samples drawn from the majority and minority classes are both set to the sample size of the minority class in the original dataset.

Random forest: sample size

	0	1
Train Data	41340	7199

	0	1
Test Data	17717	3084

```r
# model
library(randomForest)  # random forest modeling
sampsize = rep(min(as.integer(summary( the_train$quitIn30Days))),2)
emp_res_rose_RF <- randomForest(quitIn30Days ~ .,
                         data = the_train,
                         ntree=1000,sampsize=sampsize)
```

Confusion Matrix for Training Set:

```
##
##          0     1
##    0 38921   285
##    1  2419  6914
```

Confusion Matrix for Validation Set:

```
##     y_pred
##          0     1
##    0 16649  1068
##    1   140  2944
```

Method	Acc_Positive.Recall.	Acc_Negative	Precision	F_measure	G_mean	Weighted_Accuracy
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
BRF : 0.4	0.9604112	0.9414852	0.7408122	0.8364384	0.9509011	0.9509482

1 row

Method	Acc_Positive.Recall.	Acc_Negative	Precision	F_measure	G_mean
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
BRF :sampsize only	0.9546044	0.9397189	0.7337986	0.8297632	0.9471324

Random forest: sample size & cutoff = c(0.4, 0.6)

```
library(randomForest)  # random forest modeling
sampsize = rep(min(as.integer(summary( the_train$quitIn30Days))),2)
emp_res_rose_RF <- randomForest(quitIn30Days ~ .,
                           data = the_train,
                           ntree=1000,sampsize=sampsize,cutoff = c(0.4, 0.6),replace = T)
```

Confusion Matrix for Training Set:

```
        0     1
0   39971   560
1    1369  6639
```

Confusion Matrix for Validation Set:

```
   y_pred
        0     1
0   17134   583
1     293  2791
```

Method	Acc_Positive.Recall.	Acc_Negative	Precision	F_measure	G_mean	Weighted_Accuracy
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
BRF : 0.4	0.9222114	0.9668844	0.829046	0.8731505	0.9442838	0.9445479

Method	Acc_Positive.Recall.	Acc_Negative	Precision	F_measure	G_mean	Weighted_Accuracy
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
BRF : 0.4	0.9049935	0.9670938	0.8272081	0.8643543	0.9355285	0.9360436

Random forest: sample size & cutoff = c(0.45, 0.55)

```r
library(randomForest)  # random forest modeling
sampsize = rep(min(as.integer(summary( the_train$quitIn30Days))),2)
emp_res_rose_RF <- randomForest(quitIn30Days ~ .,
                           data = the_train,
                           ntree=1000,sampsize=sampsize,cutoff = c(0.45, 0.55),replace = T)
```

Confusion Matrix for Training Set:

```
       0     1
0  39556   388
1   1784  6811
```

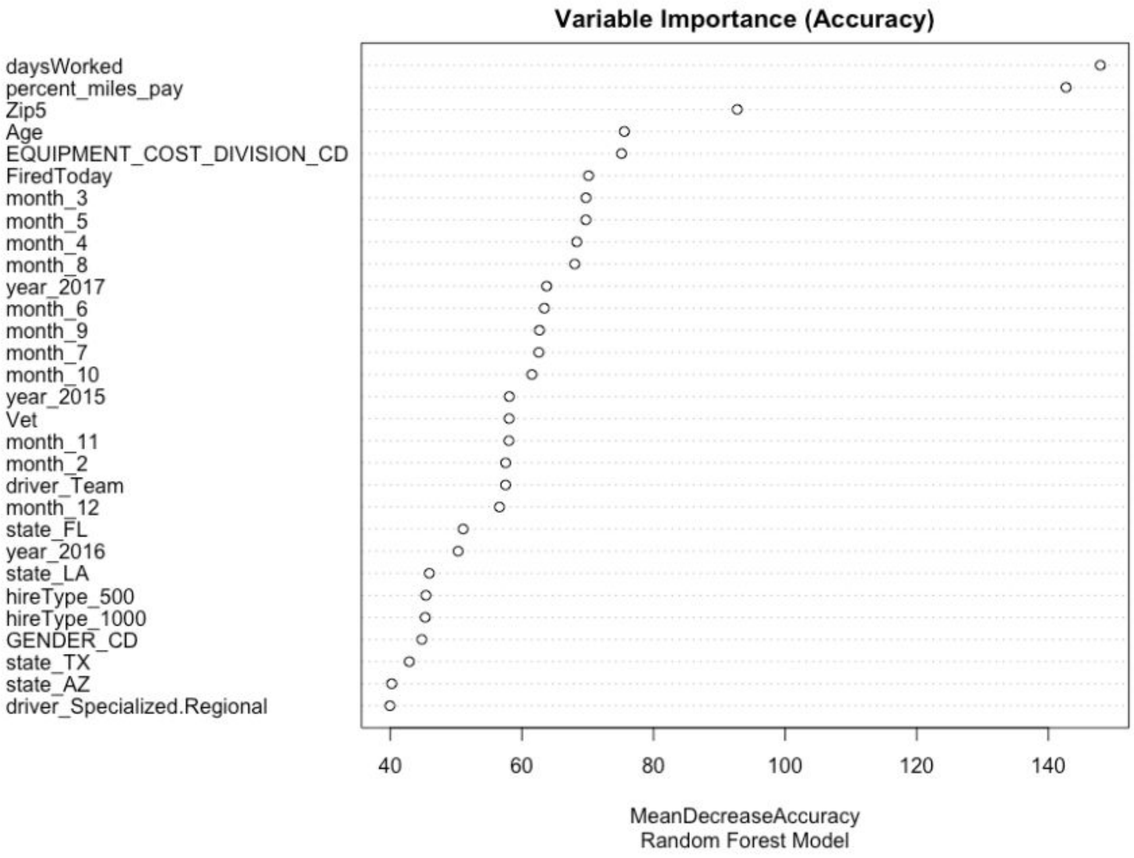
Confusion Matrix for Validation Set:

```
  y_pred
       0     1
0  16957   760
1    211  2873
```

Method	Acc_Positive.Recall.	Acc_Negative	Precision	F_measure	G_mean	Weighted_Accuracy
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
BRF : 0.4	0.9461036	0.9568457	0.7924375	0.8624794	0.9514595	0.9514746

Method	Acc_Positive.Recall.	Acc_Negative	Precision	F_measure	G_mean	Weighted_Accuracy
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
BRF : 0.45	0.9315824	0.9571033	0.7908065	0.8554414	0.9442566	0.9443429

Important Variables



Future Work

For Werner to easily put this model into action, we would eventually like to:

- Create an **application** or executable that would run this model daily/weekly/monthly to keep Werner in the know on their employee attrition