Blood Donation Prediction

ETM 538

Peter Boss
Mengyu Li
Jordan Hilton
Andey Nunes, MS

Introduction

We analyzed a data set of past blood donations to predict future donations.

Data was obtained from Driven Data, a data-analysis competition website:
https://www.drivendata.org/competitions/2/warm-up-predict-blood-donations/

The objective is to predict which donors, based on their history, are likely to give blood in the future.

Description of Data

The data comprises 576 observations on 5 variables. Each observation represents a different patient.

- Input variable: Recency = Months since last donation
- Input variable: Time = Months since first donation
- Input variable: Freq = Total donations made
- Input variable: Vol = Total volume of blood donated
- Response variable: Target = Was a donation made in March 2007

Predictions were made about whether a patient donated in March 2007

Data Example

recency	freq	vol	time	target
2	50	12500	98	1
0	13	3250	28	1
1	16	4000	35	1
2	20	5000	45	1
1	24	6000	77	0
4	4	1000	4	0

Graphical Correlation between Variables

Overview of Analysis

We performed four types of analysis on the data.

- Naive Linear Regression
- K Nearest Neighbors
- Decision Tree
- Cross-Validation Logistic Regression
- Cross-Validation Random Forest

The analysis with the best error rate was K Nearest Neighbors.

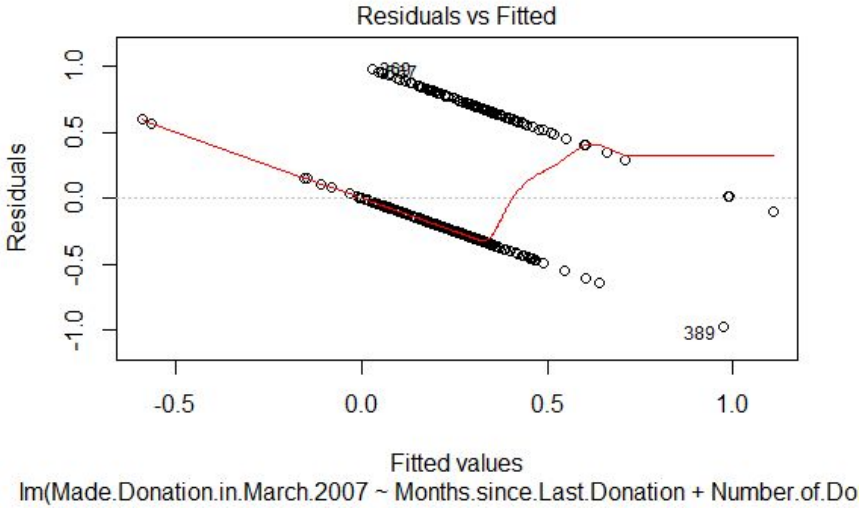
Linear Regression

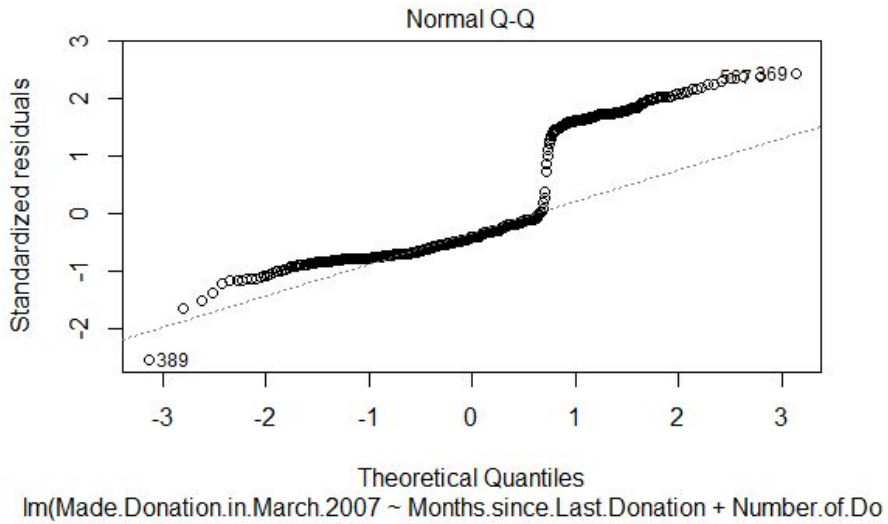
We tried ordinary linear regression as a standard first step.

The response variable is bimodal, and the residuals did not follow a normal distribution, so linear regression is not appropriate for this data set.

An analysis using logistic regression appears below.

LM: Non-Normal Residuals





K Nearest Neighbors

For this analysis we divided the training data into a training fold and a validation fold, classified each validation point by its nearest neighbors, and used that information to train weights. Our final analysis was performed using the “knn” function in the “class” library, but we did a k=1 run by hand first.

KNN: Hand Run Method

We used Euclidean distance because it made sense with our numeric variables, and because the final package used Euclidean distance. Considering only the nearest neighbor, we found the distance between every validation point and training point with the “dist” function, storing those in an intermediate table, and then found the minimum distance in every column of that table and recorded the classification of that point. This took ~100 lines of code and resulted in a 19.7% error rate.

KNN: Hand Run Results

	minimumdistance <dbl>	occurrences <int>	donation <int>
1	0.00000	1	0
2	0.00008	1	0
3	0.00016	1	1
4	0.02703	1	0
5	0.00016	1	0
6	0.00000	1	1

```{r knn errorrate}
predictedresults<-distanceresults[,3]
originaldata <- data[501:576,6]
correctanswers <- sum(predictedresults==originaldata)
errorrate_knn <- 1 - correctanswers/length(originaldata)
errorrate_knn
```

```
[1] 0.1974
```

KNN: Using the Library Is Better

```
```{r knn usinglibrary}
train<-databucket
test<-validationbucket
cl<-data[1:500,6]
libraryresults<-knn(train, test, cl, k=3, prob=TRUE)
libraryresults
```

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

k	error
1	18.4%
2	13.1%
3	10.5%
4	9.2%

4 lines of code for a 10% error rate!

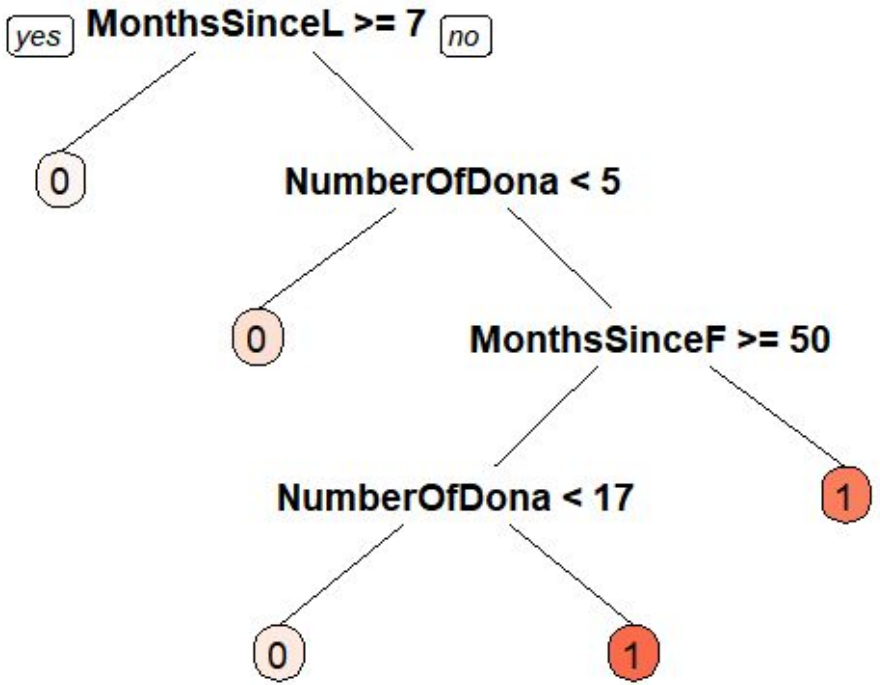
Decision Tree

Check the structure of data frame

```
'data.frame':   576 obs. of  6 variables:
 $ X                         : int  619 664 441 160 358 335 47 164 736 436 ...
 $ Months.since.Last.Donation : int  2 0 1 2 1 4 2 1 5 0 ...
 $ Number.of.Donations        : int  50 13 16 20 24 4 7 12 46 3 ...
 $ Total.Volume.Donated..c.c..: int  12500 3250 4000 5000 6000 1000 1750 3000 11500
750 ...
 $ Months.since.First.Donation: int  98 28 35 45 77 4 14 35 98 4 ...
 $ Made.Donation.in.March.2007: int  1 1 1 1 0 0 1 0 1 0 ...
```

DT: Trained Classifier Result

- **Split data into train and test data, p = 0.8**
- **Cross validation into 10 folds**
- **Cp = 0.04337671**



DT: Confusion Matrix

- **Accuracy = 0.7193**
- **The error rate = 0.2807**
- **Kappa = 0.1817, kappa coefficient is slight**

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 73 18
         1 14  9

               Accuracy : 0.7193
```

Cross-validation Logistic Regression

Specified `glm` model, family = “binomial”

$$MAE = \frac{\sum_{i=1}^{n} |Actual_i - Predicted_i|}{n}$$

Mean absolute

Error = 0.2379

foldID	MAE	accuracy	precision	recall
1	0.2069	0.7931	0.6667	0.1538
2	0.2522	0.7478	0.5000	0.0690
3	0.2957	0.7043	0.2500	0.0312
4	0.2348	0.7652	0.4000	0.0769
5	0.2000	0.8000	0.6250	0.2000

Cross-validation Random Forest

Five-fold cross validation using the *rsample*, *purrr*, and *ranger* packages

Mean Absolute Error = 0.217

Best random forest model (lowest mean MAE and best recall) used 3 variables.

mtry	rf_mean_mae	rf_mean_accuracy	rf_mean_precision	rf_mean_recall
1	0.2170	0.7830	0.5930	0.3342
2	0.2257	0.7743	0.5594	0.3631
3	0.2343	0.7657	0.5281	0.3902

Conclusion

With our different analyses, we saw these error rates.

- Linear model: NA, wrong approach for this data set
- K Nearest Neighbors: 19.7% by hand, 10.5% with the R package “class”
- Decision Tree: 28.1%
- Cross-Validation Logistic Regression: 23.8%
- Cross-Validation Random Forest: 21.7%

The most accurate model is K Nearest Neighbors. Our home-grown version is the runner-up.