Investigation of High-order Interactions in VisionZero Project

Yiqiao Yin

June 25, 2019

Table of Contents

Introduction

This project explores VisionZero Traffic Dataset, a joint project with Department of Transportation (DOT) in New York City. We are particularly motivated to investigate what may be the potential explanation for fatalities or injuries. In doing so, we deliver executable solutions for the policy makers at DOT to make better judgments for traffice in New York City.

In the context of big data such as VisionZedro project, good prediction is essential. Conventional approaches to prediction problems rely on trial and error search to evaluate models and machine learning algorithms through cross-validation may face challenges when data sets do not have sufficient sample size. The above direction may guide analysis to less-predictive variable sets due to overfitting in training set.

Data

We are looking at a variety of treatments executed by DOT in the past since 2002. The treatments are exposed in tree-based structures. For example, in streetscape elements, we may have Slow Turn Wedge interacting with Slow Turn Box as well as Right Turn Signal. We have also have Bike Lane and Raised Crosswalk together imposing an impact to the ongoing traffic.

We collect and clean up a list of these treatments and simply mark them treatment 1, 2, …, and so on. Each treatment in binary form so they are coded as 1 if there exists one and 0 otherwise.

```
# Set Working Directory
path <- "C:/Users/eagle/OneDrive/STATS GR8201 - Theo Stat [2019 Spring]/3. Vi
sionZero (Jun 2019)"
setwd(paste0(path))
```

Lab Procedure

The following section we search for an algorithm to further explore our target, Attrition, which is measured by 0 if the employee stays and 1 if the employee leaves. The task is to deliver a solution, a trainable machine, such that we can predict a new candidate's probability of Attrition with high accuracy rate.

The following let me briefly introduce different algorithms.

Bagging

Consider a regression problem. Suppose we fit a model to our training data $Z = \{x_1, y_1), \dots, (x_N, y_N)\}$, obtaiing the prediction $\hat{f}(x)$ at input x. Bootstrap aggregation or bagging averages this prediction over a collection of bootstrap samples, thereby reducing its variance. For each bootstrap sample Z^{*b} with $b = 1,2,\dots,B$,we fit out model, giving prediction $\hat{f}^{*b}(x)$. The baggting estimate is defined by

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x).$$

## Gradient Boosting Machine

To tackle a data set with Gradient Boosting Machine, we first need to define loss function using $f(x)$ to predict y on the training data, which is

$$L(f) = \sum_{i=1}^{N} L(y_i, f(x_i)).$$

Here the goal is to mimnimize $L(f)$ with respect to f while f can be trained with a sum of trees. Hence, the algorithm has the following objective function

$$\hat{f} = \underset{f}{\text{argmin}} L(f)$$

Naive Bayes

The naive Bayes model assumes given a class $G = j$, the features X_k are independent:

$$f_j(X) = \prod_{k=1}^{p} f_{jk}(X_k)$$

Then we can derive the following

$$\begin{aligned}\log\frac{\mathrm{P}(G=l|X)}{\mathrm{P}(G=J|X)} &= \log\frac{\pi_l f_l(X)}{\pi_J f_J(X)} \\ &= \log\frac{\pi_l}{\pi_J}+\sum_{k=1}^{p}\log\frac{f_{lk}(X_k)}{f_{JK}(X_k)} \\ &= \alpha_l+\sum_{k=1}^{p} g_{lk}\,(X_k)\end{aligned}$$

This has the form of a generalized additive model which can train machine to learn it.

Linear Model or Least Squares

Linear model is the most common and famous algorithm and remained mainstream of statistics for decades. Given features $X^T = (X_1, \ldots, X_p)$, we predict the output Attrition using the following model

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j\,\hat{\beta}_j.$$

In this case, we can simply write this as inner product of two matrices, i.e. $\hat{Y} = X^T\,\hat{\beta}$. We fit this model on training set and find the weights of features by picking coefficients β to minimize the following residual sum of squares (RSS)

$$\mathrm{RSS}(\beta) = \sum_{i=1}^{N}(\,y_i - x_i^T\beta)^2,$$

which is a quadratic function of the parameters and there is always a minimum (the most optimal point).

Tree-based Algorithm

Random Forest (RF), iterative Random Forest (iRF), and Bayesian Additive Regression Tree (BART) are all tree-based algorithms. To make metters simple, let us use X_1 and X_2, any two variables in a given data set, as an example. We split at $X_1 = t_1$. Then the region $X_1 \leq t_1$ is split at $X_2 = t_2$ and the region $X_1 > t_1$ is split at $X_1 = t_3$ and so on. What this does is to partition into regresions $R_1, R_2, \ldots$. The corresponding regression model predicts Y with a constant c_m in region R_m, that is,

$$\hat{f}\,(X) = \sum_{m=1}^{M} c_m\, I\{(X_1, X_2) \in R_m\}.$$

Lab Results

Let us take a look at the lab results. We summarize the important features that we are using in the machine learning algorithms. Then we look at performance comparison.

Important Features

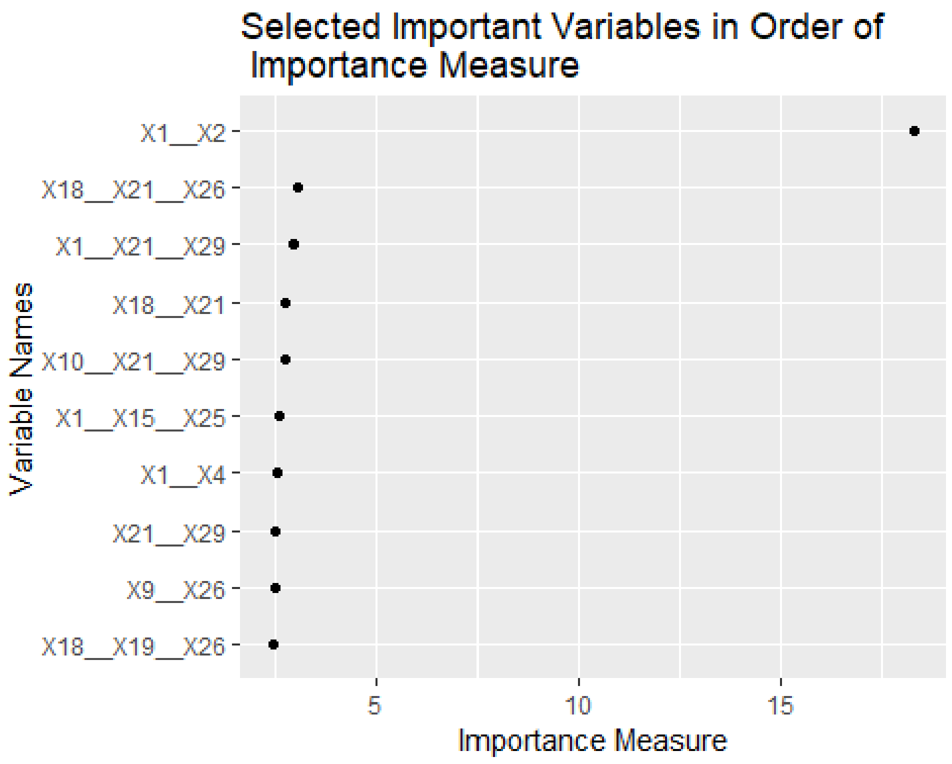
We select features using importance measure based on partitions.

```
selected_variables <- data.frame(read.csv(paste0(path, "/results/selected_var
iables_names_8_rounds_2000_from_BDA.csv")))
data.frame(selected_variables)
```

```
##     X Top_Variable_Sets Importance_Measure Variable_Names
## 1   1               1_2             18.303         X1__X2
## 2   2          18_21_26              3.048  X18__X21__X26
## 3   3           1_21_29              2.918   X1__X21__X29
## 4   4             18_21              2.743       X18__X21
## 5   5          10_21_29              2.728  X10__X21__X29
## 6   6           1_15_25              2.592   X1__X15__X25
## 7   7               1_4              2.509         X1__X4
## 8   8             21_29              2.483       X21__X29
## 9   9              9_26              2.457        X9__X26
## 10 10          18_19_26              2.422  X18__X19__X26
```

We select features using importance measure based on partitions.

```
# Visualization:
# Let us plot the important variable sets according to measures.
library(ggplot2)
ggplot(
  selected_variables,
  aes(Importance_Measure, reorder(Variable_Names, Importance_Measure))) + geo
m_point() +
  ggtitle("Selected Important Variables in Order of \n Importance Measure") +
  xlab("Importance Measure") + ylab("Variable Names")
```



Tuning and Cross-Validation

```
Final_Performance <- data.frame(read.csv(paste0(path, "/results/performance_4
_fold_used_iscore_top_11_var.csv")))
data.frame(Final_Performance)
```

```
##   X                                     Algo CV_Result
## 1 1                    Gradient Descent (GD)     0.584
## 2 2          Bagging or Bootstrap Aggregation     0.927
## 3 3                 Gradient Boosting Machine     0.902
## 4 4                               Naive Bayes     0.700
## 5 5            Linear Model or Least Squares     0.971
## 6 6                             Random Forest     0.927
## 7 7                   iterative Random Forest     0.956
## 8 8 Bayesian Additive Regression Tree (BART)     0.912
## 9 9    Deep Artificial Neural Network (DANN)     0.486
```

To get a better picture how robust an algorithm is, let us take a look at the performance on held-out test set.

```
Held_out_Test <- data.frame(read.csv(paste0(path, "/results/performance_5_fol
d_test_fold_used_iscore_top_11_var.csv")))
data.frame(Held_out_Test)
```

```
##   X                                     Algo Test_Result
## 1 1          Bagging or Bootstrap Aggregation       0.972
## 2 2                 Gradient Boosting Machine       0.903
## 3 3                               Naive Bayes       0.847
## 4 4            Linear Model or Least Squares       0.898
## 5 5                             Random Forest       0.903
## 6 6                   iterative Random Forest       0.903
## 7 7 Bayesian Additive Regression Tree (BART)       0.972
## 8 8    Deep Artificial Neural Network (DANN)       0.429
```

Conclusion

After careful investigation of VisionZero Dataset, we proposed a generalized search algorithm to detect high-order interactions in this dataset as a screening technique to target correct model specification. This allows to achieve high performance for common machine learning algorithms.