Data Science Task

Arpita Bhattacharya

15/11/2021

Aim

To identify a common point of compromise in the given dataset.

Question to be answered in this task

Which merchant is the compromised merchant? Hence, what are the dates of the common point of compromise and exploitation?

Coding environment

I used the R programming language in RStudio to find this.

Methodology to answer the question

There are two approaches to catching a fraud. The more common one is using rules and the more effective one applies machine learning. In my methodology to find the compromised merchant, I have used the rule-based approach.

First, I'll summarize the rule-based approach for fraud detection.
Fraudulent activities in finance can be detected by looking at on-surface and evident signals such as unusually large transactions. They deserve additional verification. Purely rule-based systems use algorithms that perform several fraud detection scenarios, which are manually written by fraud analysts. They find a new signal for fraud and create a rule for it.

Using this approach, the common point of compromise is found by identifying an unusually high number of transactions are observed with the same merchant. This takes place some time before fraudulent transactions were seen for those accounts, which is when the exploitation takes place.

Overview of my R code

First, the required R packages and the given dataset are imported into `transactions_dataset`.

```r
# Libraries
library(data.table)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.0     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()   masks data.table::between()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::transpose() masks data.table::transpose()
```

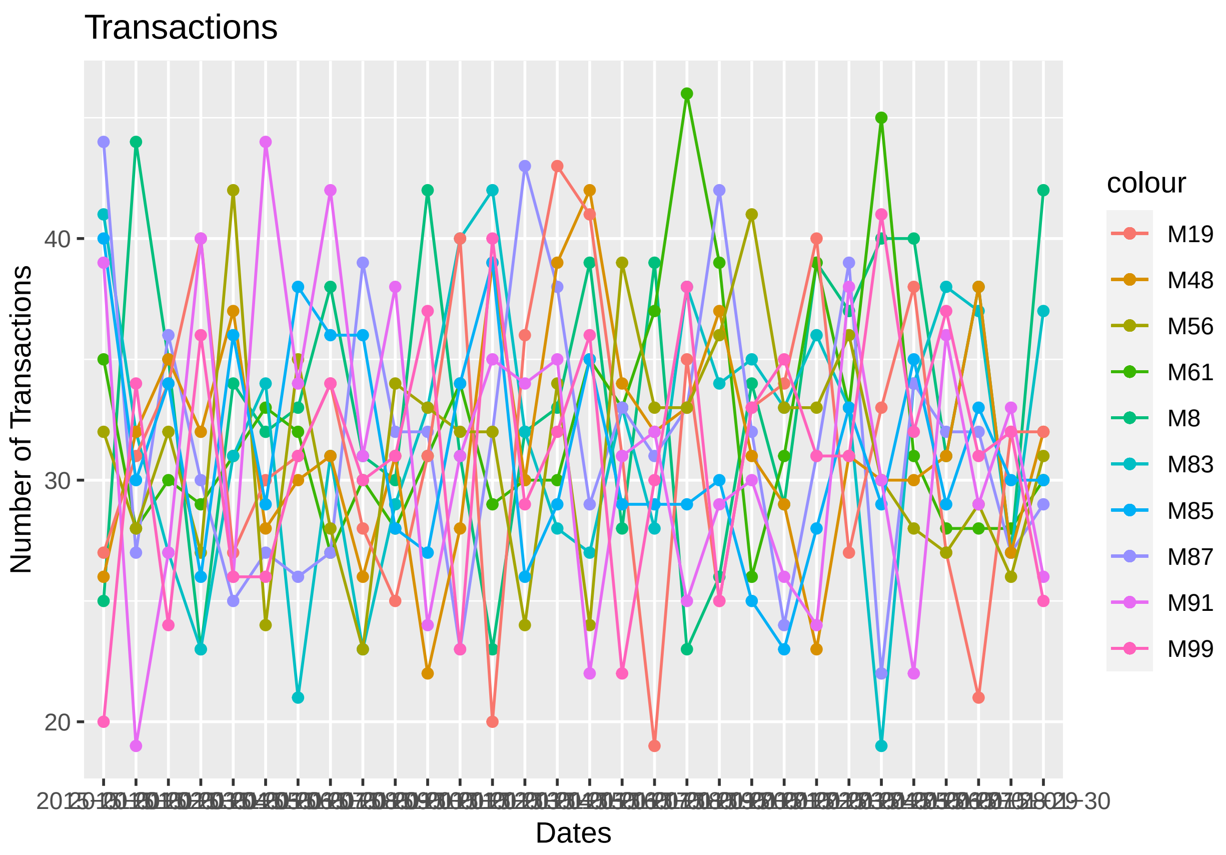
```
library(dplyr)
library(ggplot2)
library(grid)

# Importing given dataset
transactions_dataset <- read.csv("/Users/arpitabhattacharya/Desktop/Warwick /Internship/Data_Science_Ta
```

I then count the number of transactions made under each merchant, saving this in the `merchant_count` table.

```
merchant_count <- transactions_dataset %>%
    count(transactions_dataset$merchant, sort = TRUE)
names(merchant_count)[names(merchant_count) == "transactions_dataset$merchant"] <- "merchant"
```

Then, from the `merchant_count`, I take the top 10 merchants with the most number of transactions under them. For these 10 merchants, I filter out their information into different data frames, each for one of the 10 merchants (`Merchant_M**`). Next, for each merchant, I count the number of transactions on different dates and save these in different data frames, one for each merchant (`Dates_M**`). Finally, for each of the 10 merchants, I plot the number of transactions versus the dates the transactions were made.

```
for (i in 1:10) {
    Merchant <- paste("Merchant_", merchant_count$merchant[i],
        sep = "")
    merchant_nam <- filter(transactions_dataset, transactions_dataset$merchant ==
        merchant_count$merchant[i])
    assign(Merchant, merchant_nam)
    Dates <- paste("Dates_", merchant_count$merchant[i], sep = "")
    Dates_tab <- merchant_nam %>%
        count(merchant_nam$date, sort = TRUE)
    names(Dates_tab)[names(Dates_tab) == "merchant_nam$date"] <- "Dates"
    names(Dates_tab)[names(Dates_tab) == "n"] <- "Transactions"
    assign(Dates, Dates_tab)
    print(ggplot(data = Dates_tab, aes(x = Dates, y = Transactions,
        group = 1, colour = merchant_count$merchant[i])) + geom_point() +
        geom_line())
}
```



[NOTE: 1.The code above produces 10 different plots, one for each merchant. But, I have put all plots in one graph, to make it visually easier to notice my observations.
2.On the x-axis, the dates are cramped due to lack of space. It is labelled from 2015-01-01 to 2015-01-30 from left to right.]

Next, I go through all the various data frames `Merchant_M**` and `Dates_M**` using the code below. From there, I find out the merchant that had an unusually large of transactions. Merchant M61 is the one that is observed to have an unusually large of transactions, so, **merchant M61 is the compromised merchant**. And from the plot of M61, which is previously plotted, I observed this hike in transactions from 2015-01-17 to 2015-01-19. Hence, the **dates of compromise is 2015-01-17 to 2015-01-19**.

```
Dates_most = 0
for (i in 1:10) {
    merchant_nam <- filter(transactions_dataset, transactions_dataset$merchant ==
        merchant_count$merchant[i])
    Dates_tab <- merchant_nam %>%
        count(merchant_nam$date, sort = TRUE)
    if (Dates_tab[1, 2] > Dates_most) {
        Dates_most = Dates_tab[1, 2]
        Merchant_most = merchant_count$merchant[i]
        merchant_nam_most <- filter(transactions_dataset, transactions_dataset$merchant ==
            merchant_count$merchant[i])
        Dates_tab_most <- merchant_nam_most %>%
            count(merchant_nam_most$date, sort = TRUE)
    }
}
names(Dates_tab_most)[names(Dates_tab_most) == "merchant_nam_most$date"] <- "Dates"
print(Dates_most)
```

```
## [1] 46
```

```
print(Merchant_most)
```

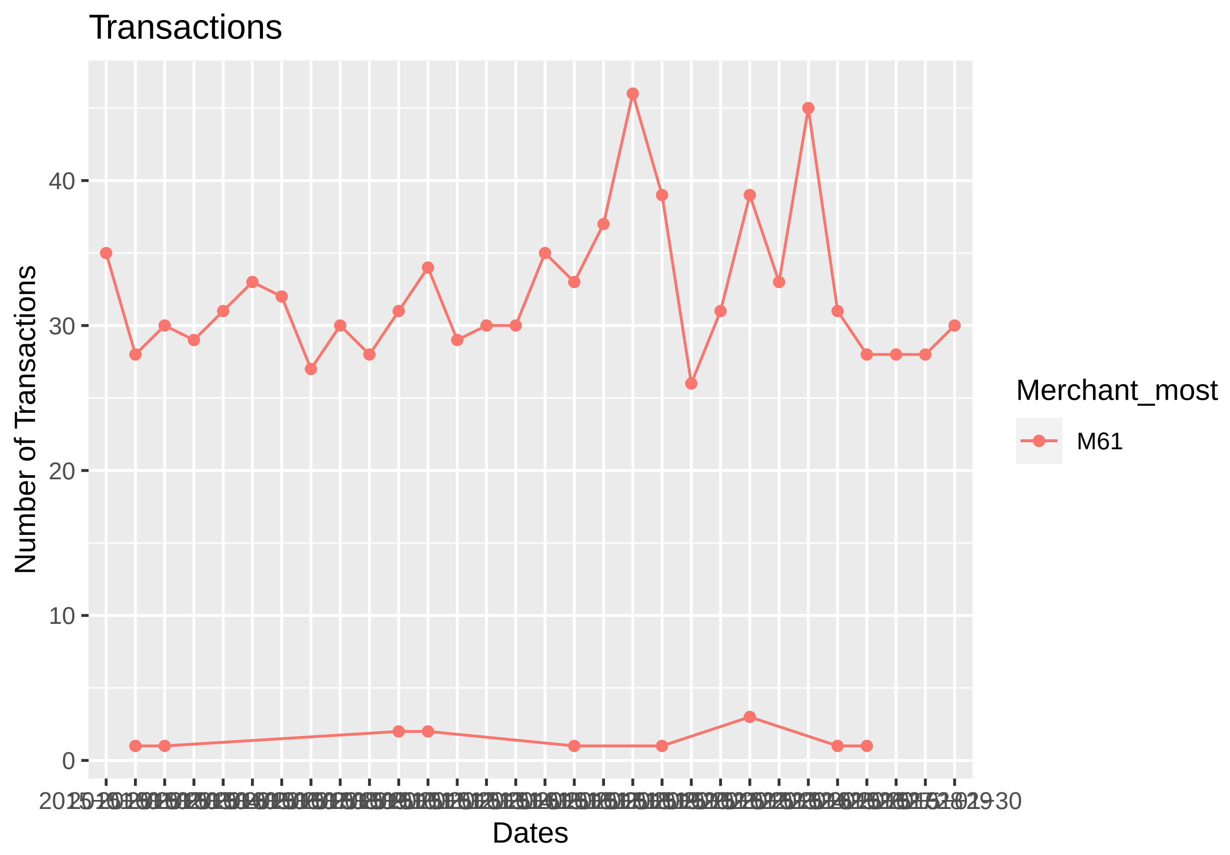
```
## [1] "M61"
```

Next, I filter the fraudulent transactions from the original dataset into `fraud_transaction`.

```
fraud_transaction <- filter(transactions_dataset, transactions_dataset$fraud ==
    "True")
```

Then, I filter the information of the merchant with the highest amount of transactions, that is merchant M61, into `FMerchant`. But, this contains the information with only the transactions that are fraudulent. From that table, I then count the number of fraudulent transactions on different dates and save them in another data frame, `FDates`. I then plot the number of transactions versus the dates and the number of fraudulent transactions versus the dates on the same graph.

```
FMerchant <- filter(fraud_transaction, fraud_transaction$merchant ==
    Merchant_most)
FDates <- FMerchant %>%
    count(FMerchant$date, sort = TRUE)
names(FDates)[names(FDates) == "FMerchant$date"] <- "Dates"

ggplot() + geom_point(data = Dates_tab_most, aes(x = Dates, y = n,
    group = 1, colour = Merchant_most)) + geom_line(data = Dates_tab_most,
    aes(x = Dates, y = n, group = 1, colour = Merchant_most)) +
    geom_point(data = FDates, aes(x = Dates, y = n, group = 1,
        colour = Merchant_most)) + geom_line(data = FDates, aes(x = Dates,
    y = n, group = 1, colour = Merchant_most)) + labs(x = "Dates",
    y = "Number of Transactions", title = "Transactions")
```



[NOTE: On the x-axis, the dates are cramped due to lack of space. It is labelled from 2015-01-01 to 2015-01-30 from left to right.]

From the above plot, I observe the hike in fraudulent transactions of M61 after the dates of compromise, that is, from 2015-01-20 to 2015-01-23. Hence, the **dates of exploitation is 2015-01-20 to 2015-01-23**.

If I had more time, how would I refine the solution?

If I had more time, first, I would try to improve this code further to find the highest number of transactions along with the **rate of the transactions** by finding the slope of the line graphs. The merchant that shows an unusually high number transactions with a high rate or frequency of transactions in a short period of time, would be the compromised merchant.

Second, with **additional data** about the transactions given, such as the location of the transaction, device used for the transaction, IP addresses, time of the day when transaction was made, etc, a better, more accurate fraud analysis can be performed.

Finally, the given task can be fulfilled by using **machine learning algorithms**. Machine learning can crunch and evaluate large amount of data in a short period of time. It is more efficient, accurate and can detect patterns in real-time. For this task we can use a combination of supervised and unsupervised techniques to detect fraudulent transactions. The following algorithms could be used to perform the fraud analysis.

1. *K-means clustering*: This algorithm would work iteratively and divide n observations in the dataset into k clusters. These data points would be clustered based on the similarity of the features in the dataset.
2. *Logistic regression*: This method would then be used to create an algorithm to predict whether a transaction is fraudulent or not.
3. *Local outlier factor*: This algorithm would be used for anomaly detection. That is, it would calculate the local density of data points and would identify regions with similar density in the data set. Then, it

would distinguish points with much lower density than other neighbours by using the locality concept. These points are the outliers (fraudulent transactions).

4. *Decision tree*: This would be used to create a set of rules that model customers' normal behavior. The dataset would be trained using examples of fraud to detect anomalies.
5. *Random forests*: This would be built by ensembling the decision trees. Ensembling multiple models is a common approach to make predictions more accurate. This would leverage the strengths of the models and would make a decision as precise as possible.
6. *Neural networks*: This technique is inspired by the way the human brain works. It would learn and adapt to the different patterns of the customers' normal behavior, identifying the fraudulent transactions, and hence, the fraud in real-time.

Results of the task

- Identity of the compromised merchant: M61
- Dates of the compromise: 2015-01-17 to 2015-01-19
- Dates of the exploitation: 2015-01-20 to 2015-01-23
- Code used to identify the common point of compromise: Github link to code
- Document to describe my methodology: Described above.