PSTAT231 Final Project: Credit Card Fraud Detection with Machine Learning

Yuer Hao

2022-11-02

Contents

"Have you gone insane? How come to earth you were doing the a Casino?!" My sweet dream was interrupt by the call (in fact yelling) of my mother. Of course I did not spent 10,000 yuan in a casino–it was credit card fraud. Studies shows (data) have once or more expereinced credit card fraud. Especially in card use overseas, credit card information is easily attainable as time differentiates in the banking systems. Although most banks usually discharge the cardholder, the money are not refunded in many cases. How can we use our acknowledge from PSTAT231 to solve this real world problem?

```
knitr::include_graphics("/Users/Yuer_Hao/Desktop/PSTAT231---Final-Project/Picture/Cover.jpg")
```

1. Introduction

1.1 The purpose of this project

1.2 Some facts you need to know about Credit Card Fraud

Almost 40 percent of card holders do not have email or text alerts from their credit card company or bank enabled. Around 81 percent of victims without these notifications had to take additional action to reverse fraudulent charges, compared to just 19 percent of those with alerts enabled.

1.3 Why might this model be useful?

2. An overview of dataset

This project uses MACHINE LEARNING GROUP - ULB's dataset from Kaggle

The dataset contains credit card transactions done by European cardholders in September 2013. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is quite unbalanced, with frauds making up 0.172% of all transactions in the positive class (frauds) account.
There are 284807 observations and 31 columns in this dataset. There are 1 response variable and 30 predictor variables. Additionally, 30 of them are numerical, while 1 is binary. The response variable, "Class," has a value of 1 in cases of fraud and 0 in all other cases.

- `Time`: (Data Type: continuous) Number of seconds elapsed between this transaction and the first transaction in the dataset
- `V1-V28`: (Data Type: continuous) May be result of a PCA Dimensionality reduction to protect user identities and sensitive features
- `Amount`: (Data Type: continuous) Transaction Amount
- `Class`: (Data Type: nominal) The response variable and it takes value 1 in case of fraud and 0 otherwise
  Note: a full copy of the codebook is available in zipped final projecct files.

2.1 Loading Data and Packages

```
# read in the data
raw_data <- read.csv("/Users/Yuer_Hao/Desktop/PSTAT231---Final-Project/data/creditcard.csv")
head(raw_data)
```

```
##   Time         V1          V2        V3         V4          V5          V6
## 1    0 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778
## 2    0  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081
## 3    1 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938
## 4    1 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317
## 5    2 -1.1582331  0.87773675 1.5487178  0.4030339 -0.40719338  0.09592146
## 6    2 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755
##            V7          V8         V9         V10        V11         V12
## 1  0.23959855  0.09869790  0.3637870  0.09079417 -0.5515995 -0.61780086
## 2 -0.07880298  0.08510165 -0.2554251 -0.16697441  1.6127267  1.06523531
## 3  0.79146096  0.24767579 -1.5146543  0.20764287  0.6245015  0.06608369
## 4  0.23760894  0.37743587 -1.3870241 -0.05495192 -0.2264873  0.17822823
## 5  0.59294075 -0.27053268  0.8177393  0.75307443 -0.8228429  0.53819555
## 6  0.47620095  0.26031433 -0.5686714 -0.37140720  1.3412620  0.35989384
##          V13        V14        V15        V16         V17         V18
## 1 -0.9913898 -0.3111694  1.4681770 -0.4704005  0.20797124  0.02579058
## 2  0.4890950 -0.1437723  0.6355581  0.4639170 -0.11480466 -0.18336127
## 3  0.7172927 -0.1659459  2.3458649 -2.8900832  1.10996938 -0.12135931
## 4  0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279  1.96577500
## 5  1.3458516 -1.1196698  0.1751211 -0.4514492 -0.23703324 -0.03819479
## 6 -0.3580907 -0.1371337  0.5176168  0.4017259 -0.05813282  0.06865315
```

```
##           V19         V20          V21          V22         V23         V24
## 1  0.40399296  0.25141210 -0.018306778  0.277837576 -0.11047391  0.06692807
## 2 -0.14578304 -0.06908314 -0.225775248 -0.638671953  0.10128802 -0.33984648
## 3 -2.26185710  0.52497973  0.247998153  0.771679402  0.90941226 -0.68928096
## 4 -1.23262197 -0.20803778 -0.108300452  0.005273597 -0.19032052 -1.17557533
## 5  0.80348692  0.40854236 -0.009430697  0.798278495 -0.13745808  0.14126698
## 6 -0.03319379  0.08496767 -0.208253515 -0.559824796 -0.02639767 -0.37142658
##          V25        V26          V27         V28 Amount Class
## 1  0.1285394 -0.1891148  0.133558377 -0.02105305 149.62     0
## 2  0.1671704  0.1258945 -0.008983099  0.01472417   2.69     0
## 3 -0.3276418 -0.1390966 -0.055352794 -0.05975184 378.66     0
## 4  0.6473760 -0.2219288  0.062722849  0.06145763 123.50     0
## 5 -0.2060096  0.5022922  0.219422230  0.21515315  69.99     0
## 6 -0.2327938  0.1059148  0.253844225  0.08108026   3.67     0
```

3. Data Cleaning

While the data set that was downloaded was tidy, a few different cleaning steps were necessary before the split occurred: ### 3.1 Clean name

```
occard_data <- raw_data %>%
  clean_names()
```

3.2 Deal with imbalanced problems

Let's now determine whether or not our response variable is balanced. If not, we must resolve the situation.

```
table(occard_data$class)
```

```
##
##      0      1
## 284315    492
```

We can tell our response variable is highly unbalanced. Observations on "0" class are far more frequent than "1" class. We need to use some functions to address this problem, otherwise this will have a significant impact on our prediction models. The TA suggests that we employ the ovun.sample() function to processing it.

```
ccard_data<- ovun.sample(class~.,data = occard_data,
                         p=0.5,seed = 1,method = "under")$data
```

```
table(ccard_data$class)
```

```
##
##   0   1
## 474 492
```

our response variable is almost balanced.
3.3 Convert class to factor

```
ccard_data <- ccard_data %>%
  mutate(class = factor(class, levels = c("1", "0")))
```

3.4 Check missing value

```
sum(is.na(ccard_data))
```

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

3.5 Summary

```
summary(ccard_data$amount)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    1.29   17.53  110.80   99.99 3889.00
```

```
var(ccard_data$amount)
```

```
## [1] 79239.91
```

```
# show me how many observations in the new dataset
# show me how many variables in the new dataset
dim(ccard_data)
```

```
## [1] 966  31
```

3.6 Clean name

```
ccard_data$amount <- scale(ccard_data$amount)
head(ccard_data)
```

```
##     time        v1         v2         v3         v4          v5         v6
## 1 115330  1.948495 -2.4328242 -1.2169303 -2.4530214  0.09082679  4.2203172
## 2  97565  1.948707 -0.3391218 -1.1546341  0.3158238  0.05294384 -0.3824387
## 3 146840 -1.057784  1.9237784 -2.7290233  0.1277136  3.20514632  3.8164145
## 4 166797  1.881484 -0.3834949 -0.1647898  0.4635760 -0.88000644 -1.0167097
## 5  44774 -3.907550 -2.9204593  0.1326738  4.0886801  2.37190176 -0.7836577
## 6 151254  1.925416  0.5361020 -0.9851712  3.4202293  1.00790735  0.8207051
##            v7         v8         v9         v10         v11         v12
## 1 -2.53435629  1.1362694 -0.5960961  1.44381915 -0.32070338 -0.68661365
## 2 -0.09716884 -0.2274915  2.1491916 -0.43094995  1.22054027 -1.25835557
## 3 -0.64081125 -1.3739571 -1.2845466 -1.29969929 -0.16352126 -0.00419732
## 4 -0.28491272 -0.2206464  0.8159585 -0.05637091 -0.19202761  1.16516206
## 5 -0.87404625 -3.0750082 -1.0801413  0.82482066 -0.99749955  0.38588516
## 6  0.12124421  0.1533321 -1.1814624  1.59252107 -0.03725105  0.23320675
```

```
##          v13         v14         v15         v16          v17         v18
## 1  0.6358285 -0.82113376  0.89893329  0.09423038  0.068265414  0.5573523
## 2  1.8643940  1.59625369 -1.87460480 -0.35818920  0.246080193  0.4489676
## 3 -0.7011774 -0.09235227  0.09799396 -0.44779961  1.519808132  0.3427721
## 4  1.2091746 -0.24966393  0.36126522  0.23841666 -0.407593659 -0.7771290
## 5  1.1415790  0.12490179  0.36735428 -0.18305190  0.006499662 -0.1162191
## 6 -0.1532594  0.45743574 -1.42886978  1.13799876 -1.163919739 -0.1460211
##          v19         v20         v21        v22         v23         v24
## 1 -0.7499656 -0.05966340  0.14479793  0.5715729  0.19591337  0.70263854
## 2  0.5207497 -0.14488231 -0.01757742  0.4046979 -0.07109144 -0.47944447
## 3  0.3092393  0.68333667 -1.19656940  0.6727016 -0.04016785  0.57270376
## 4 -0.2185803 -0.03205528 -0.19212958 -0.5048663  0.43783263  0.48215724
## 5  1.2106120  0.43153187 -1.53995168  0.8945958  1.38531492 -0.02604194
## 6 -0.8563276 -0.25214395 -0.26540584 -0.8351840  0.30784382 -0.05682810
##           v25         v26         v27          v28     amount class
## 1 -0.54009091 -0.08365089  0.07641251 -0.009086192  0.1268249     0
## 2  0.20429913  0.15962962 -0.06419000 -0.074708708 -0.2060397     0
## 3  0.02569644 -0.37054060  0.33862279  0.215431992 -0.3250468     0
## 4 -0.68266256  0.19525195 -0.04039997 -0.028229323 -0.1913326     0
## 5 -0.14995883  0.27862889  0.97432304 -0.309421848  0.9046338     0
## 6 -0.20409834 -0.35081119 -0.04464170 -0.053214199 -0.3686354     0
```

We completed the the process of data cleaning.