edX HarvardX: PH125.8x | Capstone Project

Nicole Yong

9/6/2019

INTRODUCTION

This Credit Card Fraud Detection project is created to fulfil the coursework requirements of the capstone module in edX HarvardX: PH125.9x (Data Science Professional Certificate).

Credit Card Fraud Detection Dataset

This dataset contains transactions made by credit cards over two days in September 2013 by European cardholders. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

Variables:

- V1, V2, ... V28: principal components obtained with PCA
- 'Time': the seconds elapsed between each transaction and the first transaction in the dataset
- 'Amount': the transaction amount
- 'Class': the response variable; value 1 in the case of fraud and 0 otherwise

The dataset can be found on Kaggle in the following link:

- https://www.kaggle.com/mlg-ulb/creditcardfraud/

Goal of Credit Card Fraud Detection Project

The goal of the credit card fraud detection project is to identify fraudulent credit card transactions.

Measurement used: Area Under the Precision-Recall Curve

As the dataset is highly imbalanced, the accuracy will be measured using the Area Under the Precision-Recall Curve (AUPRC). The AUPRC is a single number summary of the information in the precision-recall (PR) curve.

Key Steps Performed

1. Importing data

- Downloading data from the link
- Importing data into RStudio

2. Data visualization

- Explore variables in the dataset
- Draw insights from variables

3. Data preprocessing

- Convert target variable into factor
- Rescale variables as variables with higher values may dominate algorithms

4. Spliting data into train and test dataset

- Baseline train and test dataset without additional processing
- Oversampled train set: Majority Weighted Minority Oversampling Technique (MWMOTE) adjusted train dataset
- Undersampled train set: Synthetic Minority Over-sampling Technique (SMOTE) adjusted train dataset

5. Model selection and training

- Boosted classification trees

METHODS & ANALYSIS OF CREDIT CARD FRAUD DETECTION DATASET

Load Libraries

- Load the Required Libraries
- Install Missing Packages Automatically

```
## Loading required package: tidyverse

## -- Attaching packages ---------------------------------------------------------------------- tidyvers

## v ggplot2 3.2.0     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------------------------------------------------ tidyverse_confl
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose

## Loading required package: lubridate

##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday,
##     week, yday, year

## The following object is masked from 'package:base':
##
##     date

## Loading required package: ggthemes

## Loading required package: PRROC

## Loading required package: xgboost

##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:tidyr':
##
##     expand

## Loading required package: gridExtra

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine

## Loading required package: class

## Loading required package: imbalance

## Loading required package: smotefamily

## Loading required package: ROSE

## Loaded ROSE 0.0-3

##
## Attaching package: 'ROSE'

## The following object is masked from 'package:PRROC':
##
##     roc.curve
```

Print Session Information

To help with code reproducibility, print version information about R, the OS and attached or loaded packages.

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18362)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_Singapore.1252  LC_CTYPE=English_Singapore.1252
## [3] LC_MONETARY=English_Singapore.1252 LC_NUMERIC=C
## [5] LC_TIME=English_Singapore.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] ROSE_0.0-3        smotefamily_1.3.1 imbalance_1.0.0
##  [4] class_7.3-15      gridExtra_2.3     Matrix_1.2-17
##  [7] xgboost_0.90.0.2  PRROC_1.3.1       ggthemes_4.2.0
## [10] lubridate_1.7.4   data.table_1.12.2 caret_6.0-84
## [13] lattice_0.20-38   forcats_0.4.0     stringr_1.4.0
## [16] dplyr_0.8.3       purrr_0.3.2       readr_1.3.1
## [19] tidyr_0.8.3       tibble_2.1.3      ggplot2_3.2.0
## [22] tidyverse_1.2.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.1         assertthat_0.2.1   zeallot_0.1.0
##  [4] digest_0.6.20      ipred_0.9-9        foreach_1.4.4
##  [7] R6_2.4.0           cellranger_1.1.0   plyr_1.8.4
## [10] backports_1.1.4    stats4_3.6.1       evaluate_0.14
## [13] httr_1.4.0         pillar_1.4.2       rlang_0.4.0
## [16] lazyeval_0.2.2     readxl_1.3.1       rstudioapi_0.10
## [19] rpart_4.1-15       rmarkdown_1.14     splines_3.6.1
## [22] gower_0.2.1        munsell_0.5.0      broom_0.5.2
## [25] compiler_3.6.1     modelr_0.1.4       xfun_0.9
## [28] pkgconfig_2.0.2    htmltools_0.3.6    nnet_7.3-12
## [31] tidyselect_0.2.5   prodlim_2018.04.18 codetools_0.2-16
## [34] crayon_1.3.4       withr_2.1.2        ModelMetrics_1.2.2
## [37] MASS_7.3-51.4      recipes_0.1.6      grid_3.6.1
## [40] nlme_3.1-140       jsonlite_1.6       gtable_0.3.0
## [43] magrittr_1.5       scales_1.0.0       cli_1.1.0
## [46] stringi_1.4.3      reshape2_1.4.3     timeDate_3043.102
## [49] xml2_1.2.0         vctrs_0.2.0        generics_0.0.2
## [52] lava_1.6.5         iterators_1.0.10   tools_3.6.1
## [55] glue_1.3.1         hms_0.5.0          survival_2.44-1.1
## [58] yaml_2.2.0         colorspace_1.4-1   rvest_0.3.4
## [61] knitr_1.23         haven_2.1.1
```

STEP 1: DATA GATHERING & LOADING THE DATASET

Load Credit Card Fraud Detection Dataset

```
data <- read_csv("./input/creditcard.csv")
```

Sample of dataset

```
## # A tibble: 6 x 31
##    Time     V1      V2    V3     V4      V5      V6      V7      V8     V9
##   <dbl>  <dbl>   <dbl> <dbl>  <dbl>   <dbl>   <dbl>   <dbl>   <dbl>  <dbl>
## 1     0 -1.36  -0.0728 2.54   1.38  -0.338   0.462   0.240   0.0987  0.364
## 2     0  1.19   0.266  0.166  0.448  0.0600 -0.0824 -0.0788  0.0851 -0.255
## 3     1 -1.36  -1.34   1.77   0.380 -0.503   1.80    0.791   0.248  -1.51
## 4     1 -0.966 -0.185  1.79  -0.863 -0.0103  1.25    0.238   0.377  -1.39
## 5     2 -1.16   0.878  1.55   0.403 -0.407   0.0959  0.593  -0.271   0.818
## 6     2 -0.426  0.961  1.14  -0.168  0.421  -0.0297  0.476   0.260  -0.569
## # ... with 21 more variables: V10 <dbl>, V11 <dbl>, V12 <dbl>, V13 <dbl>,
## #   V14 <dbl>, V15 <dbl>, V16 <dbl>, V17 <dbl>, V18 <dbl>, V19 <dbl>,
## #   V20 <dbl>, V21 <dbl>, V22 <dbl>, V23 <dbl>, V24 <dbl>, V25 <dbl>,
## #   V26 <dbl>, V27 <dbl>, V28 <dbl>, Amount <dbl>, Class <dbl>
```

STEP 2: DATA EXPLORATION & VISUALIZATION

Part 2.1: Inspect the dataset

Summary of dataset

```
##       Time              V1                  V2
##  Min.   :     0   Min.   :-56.40751   Min.   :-72.71573
##  1st Qu.: 54202   1st Qu.: -0.92037   1st Qu.: -0.59855
##  Median : 84692   Median :  0.01811   Median :  0.06549
##  Mean   : 94814   Mean   :  0.00000   Mean   :  0.00000
##  3rd Qu.:139321   3rd Qu.:  1.31564   3rd Qu.:  0.80372
##  Max.   :172792   Max.   :  2.45493   Max.   : 22.05773
##        V3                V4                 V5
##  Min.   :-48.3256   Min.   :-5.68317   Min.   :-113.74331
##  1st Qu.: -0.8904   1st Qu.:-0.84864   1st Qu.:  -0.69160
##  Median :  0.1799   Median :-0.01985   Median :  -0.05434
##  Mean   :  0.0000   Mean   : 0.00000   Mean   :   0.00000
##  3rd Qu.:  1.0272   3rd Qu.: 0.74334   3rd Qu.:   0.61193
##  Max.   :  9.3826   Max.   :16.87534   Max.   :  34.80167
##        V6                V7                 V8
##  Min.   :-26.1605   Min.   :-43.5572   Min.   :-73.21672
##  1st Qu.: -0.7683   1st Qu.: -0.5541   1st Qu.: -0.20863
##  Median : -0.2742   Median :  0.0401   Median :  0.02236
##  Mean   :  0.0000   Mean   :  0.0000   Mean   :  0.00000
##  3rd Qu.:  0.3986   3rd Qu.:  0.5704   3rd Qu.:  0.32735
##  Max.   : 73.3016   Max.   :120.5895   Max.   : 20.00721
##        V9                 V10                 V11
##  Min.   :-13.43407   Min.   :-24.58826   Min.   :-4.79747
##  1st Qu.: -0.64310   1st Qu.: -0.53543   1st Qu.:-0.76249
##  Median : -0.05143   Median : -0.09292   Median :-0.03276
##  Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.00000
##  3rd Qu.:  0.59714   3rd Qu.:  0.45392   3rd Qu.: 0.73959
##  Max.   : 15.59500   Max.   : 23.74514   Max.   :12.01891
##       V12                V13                V14
##  Min.   :-18.6837   Min.   :-5.79188   Min.   :-19.2143
##  1st Qu.: -0.4056   1st Qu.:-0.64854   1st Qu.: -0.4256
##  Median :  0.1400   Median :-0.01357   Median :  0.0506
##  Mean   :  0.0000   Mean   : 0.00000   Mean   :  0.0000
##  3rd Qu.:  0.6182   3rd Qu.: 0.66251   3rd Qu.:  0.4931
##  Max.   :  7.8484   Max.   : 7.12688   Max.   : 10.5268
##       V15                V16                 V17
##  Min.   :-4.49894   Min.   :-14.12985   Min.   :-25.16280
##  1st Qu.:-0.58288   1st Qu.: -0.46804   1st Qu.: -0.48375
##  Median : 0.04807   Median :  0.06641   Median : -0.06568
##  Mean   : 0.00000   Mean   :  0.00000   Mean   :  0.00000
##  3rd Qu.: 0.64882   3rd Qu.:  0.52330   3rd Qu.:  0.39968
##  Max.   : 8.87774   Max.   : 17.31511   Max.   :  9.25353
##       V18                 V19                 V20
##  Min.   :-9.498746   Min.   :-7.213527   Min.   :-54.49772
##  1st Qu.:-0.498850   1st Qu.:-0.456299   1st Qu.: -0.21172
##  Median :-0.003636   Median : 0.003735   Median : -0.06248
##  Mean   : 0.000000   Mean   : 0.000000   Mean   :  0.00000
##  3rd Qu.: 0.500807   3rd Qu.: 0.458949   3rd Qu.:  0.13304
```

```
##  Max.   : 5.041069   Max.   : 5.591971   Max.   : 39.42090
##       V21                V22                 V23
##  Min.   :-34.83038   Min.   :-10.933144   Min.   :-44.80774
##  1st Qu.: -0.22839   1st Qu.: -0.542350   1st Qu.: -0.16185
##  Median : -0.02945   Median :  0.006782   Median : -0.01119
##  Mean   :  0.00000   Mean   :  0.000000   Mean   :  0.00000
##  3rd Qu.:  0.18638   3rd Qu.:  0.528554   3rd Qu.:  0.14764
##  Max.   : 27.20284   Max.   : 10.503090   Max.   : 22.52841
##       V24                V25                 V26
##  Min.   :-2.83663   Min.   :-10.29540   Min.   :-2.60455
##  1st Qu.:-0.35459   1st Qu.: -0.31715   1st Qu.:-0.32698
##  Median : 0.04098   Median :  0.01659   Median :-0.05214
##  Mean   : 0.00000   Mean   :  0.00000   Mean   : 0.00000
##  3rd Qu.: 0.43953   3rd Qu.:  0.35072   3rd Qu.: 0.24095
##  Max.   : 4.58455   Max.   :  7.51959   Max.   : 3.51735
##       V27                V28                Amount
##  Min.   :-22.565679   Min.   :-15.43008   Min.   :    0.00
##  1st Qu.: -0.070840   1st Qu.: -0.05296   1st Qu.:    5.60
##  Median :  0.001342   Median :  0.01124   Median :   22.00
##  Mean   :  0.000000   Mean   :  0.00000   Mean   :   88.35
##  3rd Qu.:  0.091045   3rd Qu.:  0.07828   3rd Qu.:   77.17
##  Max.   : 31.612198   Max.   : 33.84781   Max.   :25691.16
##      Class
##  Min.   :0.000000
##  1st Qu.:0.000000
##  Median :0.000000
##  Mean   :0.001728
##  3rd Qu.:0.000000
##  Max.   :1.000000
```

Glimpse the dataset

```
## Observations: 284,807
## Variables: 31
## $ Time   <dbl> 0, 0, 1, 1, 2, 2, 4, 7, 7, 9, 10, 10, 10, 11, 12, 12, 1...
## $ V1     <dbl> -1.3598071, 1.1918571, -1.3583541, -0.9662717, -1.15823...
## $ V2     <dbl> -0.07278117, 0.26615071, -1.34016307, -0.18522601, 0.87...
## $ V3     <dbl> 2.53634674, 0.16648011, 1.77320934, 1.79299334, 1.54871...
## $ V4     <dbl> 1.37815522, 0.44815408, 0.37977959, -0.86329128, 0.4030...
## $ V5     <dbl> -0.33832077, 0.06001765, -0.50319813, -0.01030888, -0.4...
## $ V6     <dbl> 0.46238778, -0.08236081, 1.80049938, 1.24720317, 0.0959...
## $ V7     <dbl> 0.239598554, -0.078802983, 0.791460956, 0.237608940, 0....
## $ V8     <dbl> 0.098697901, 0.085101655, 0.247675787, 0.377435875, -0....
## $ V9     <dbl> 0.3637870, -0.2554251, -1.5146543, -1.3870241, 0.817739...
## $ V10    <dbl> 0.09079417, -0.16697441, 0.20764287, -0.05495192, 0.753...
## $ V11    <dbl> -0.55159953, 1.61272666, 0.62450146, -0.22648726, -0.82...
## $ V12    <dbl> -0.61780086, 1.06523531, 0.06608369, 0.17822823, 0.5381...
## $ V13    <dbl> -0.99138985, 0.48909502, 0.71729273, 0.50775687, 1.3458...
## $ V14    <dbl> -0.31116935, -0.14377230, -0.16594592, -0.28792375, -1....
## $ V15    <dbl> 1.468176972, 0.635558093, 2.345864949, -0.631418118, 0....
## $ V16    <dbl> -0.47040053, 0.46391704, -2.89008319, -1.05964725, -0.4...
## $ V17    <dbl> 0.207971242, -0.114804663, 1.109969379, -0.684092786, -...
## $ V18    <dbl> 0.02579058, -0.18336127, -0.12135931, 1.96577500, -0.03...
## $ V19    <dbl> 0.40399296, -0.14578304, -2.26185710, -1.23262197, 0.80...
```

```
## $ V20    <dbl> 0.25141210, -0.06908314, 0.52497973, -0.20803778, 0.408...
## $ V21    <dbl> -0.018306778, -0.225775248, 0.247998153, -0.108300452, ...
## $ V22    <dbl> 0.277837576, -0.638671953, 0.771679402, 0.005273597, 0....
## $ V23    <dbl> -0.110473910, 0.101288021, 0.909412262, -0.190320519, -...
## $ V24    <dbl> 0.06692807, -0.33984648, -0.68928096, -1.17557533, 0.14...
## $ V25    <dbl> 0.12853936, 0.16717040, -0.32764183, 0.64737603, -0.206...
## $ V26    <dbl> -0.18911484, 0.12589453, -0.13909657, -0.22192884, 0.50...
## $ V27    <dbl> 0.133558377, -0.008983099, -0.055352794, 0.062722849, 0...
## $ V28    <dbl> -0.021053053, 0.014724169, -0.059751841, 0.061457629, 0...
## $ Amount <dbl> 149.62, 2.69, 378.66, 123.50, 69.99, 3.67, 4.99, 40.80,...
## $ Class  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
```

Variable names

```
##  [1] "Time"   "V1"     "V2"     "V3"     "V4"     "V5"     "V6"
##  [8] "V7"     "V8"     "V9"     "V10"    "V11"    "V12"    "V13"
## [15] "V14"    "V15"    "V16"    "V17"    "V18"    "V19"    "V20"
## [22] "V21"    "V22"    "V23"    "V24"    "V25"    "V26"    "V27"
## [29] "V28"    "Amount" "Class"
```

Check dimension (number of rows and columns)

```
dim(data)
```

```
## [1] 284807     31
```

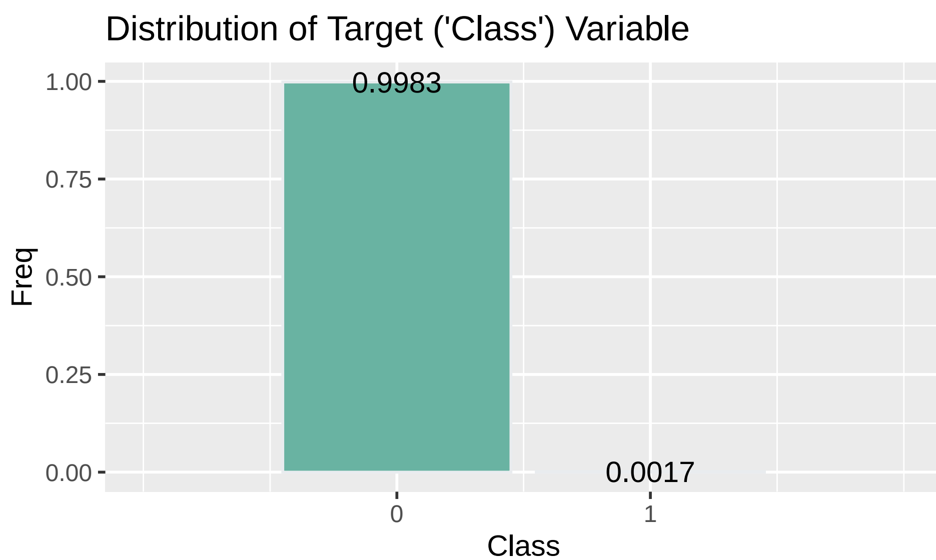
Insights from Part 2.1

1. No missing data from summary table
2. All variables are masked, except ‘Amount’, ‘Time’ and ‘Class’

Part 2.2: Inspect target variable - 'Class'

```r
table(data$Class)
```

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



Convert target variable ('Class') into factor

```r
data$Class <- factor(ifelse(data$Class == 0, "zero", "one"))
```

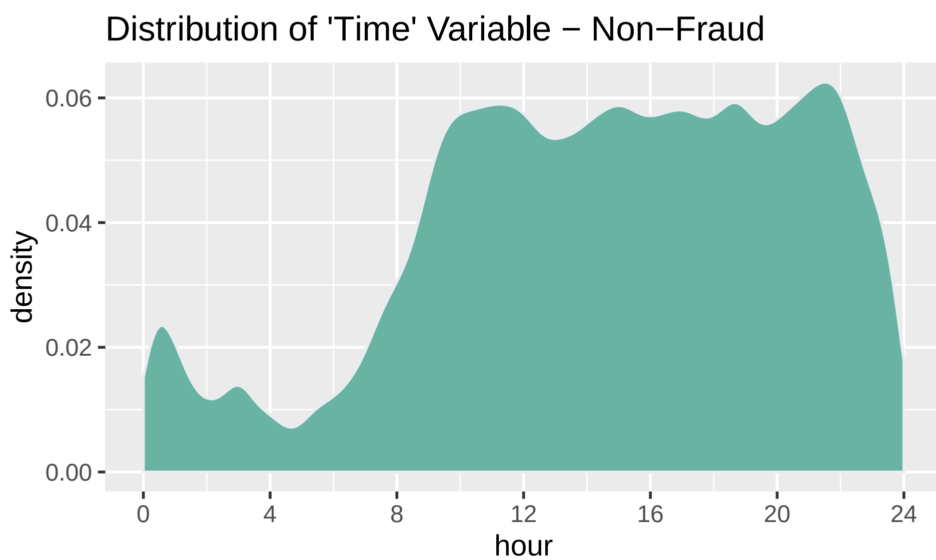
Insights from Part 2.2

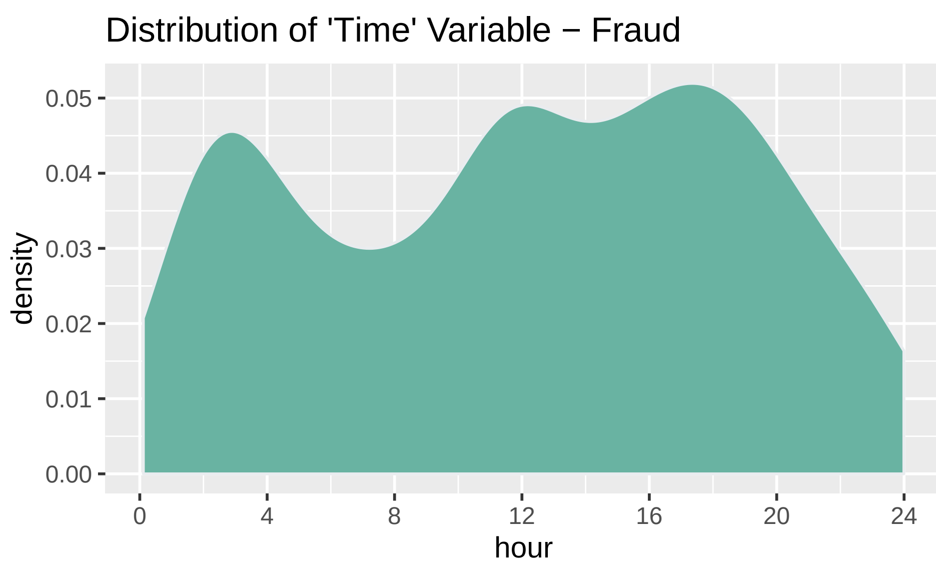
1. Highly imbalanced dataset

Part 2.3: Inspect 'Time' variable

Convert seconds to hours of the day

```
data$hour <- (data$Time/3600) %% 24
```

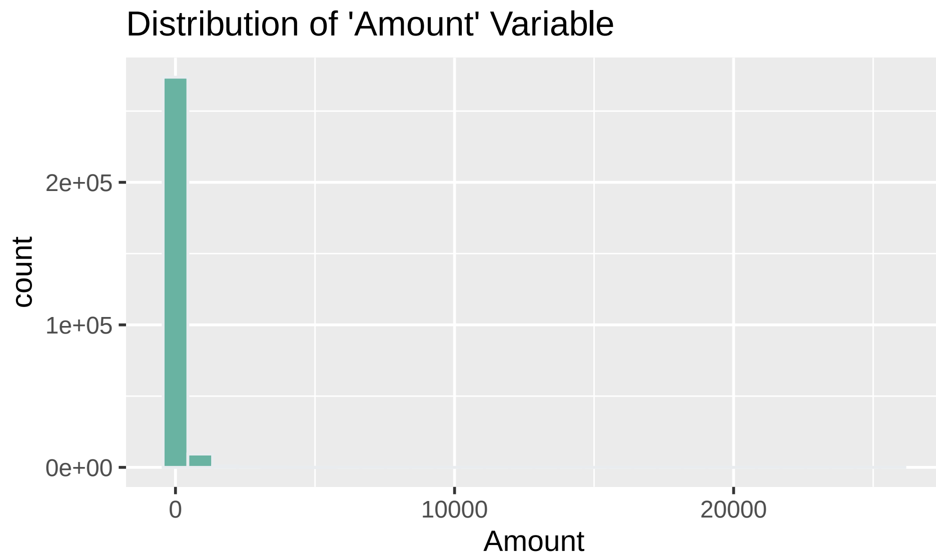


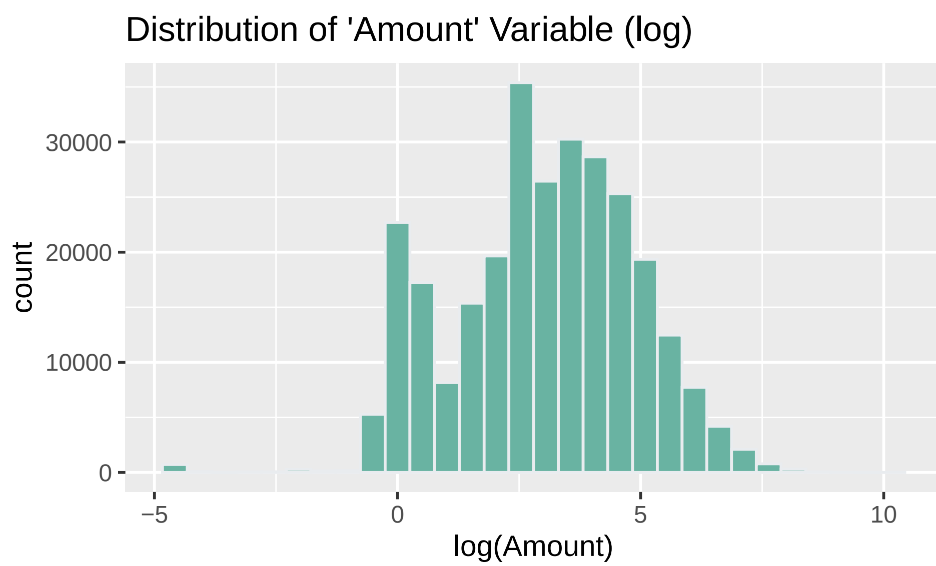


Insights from Part 2.3

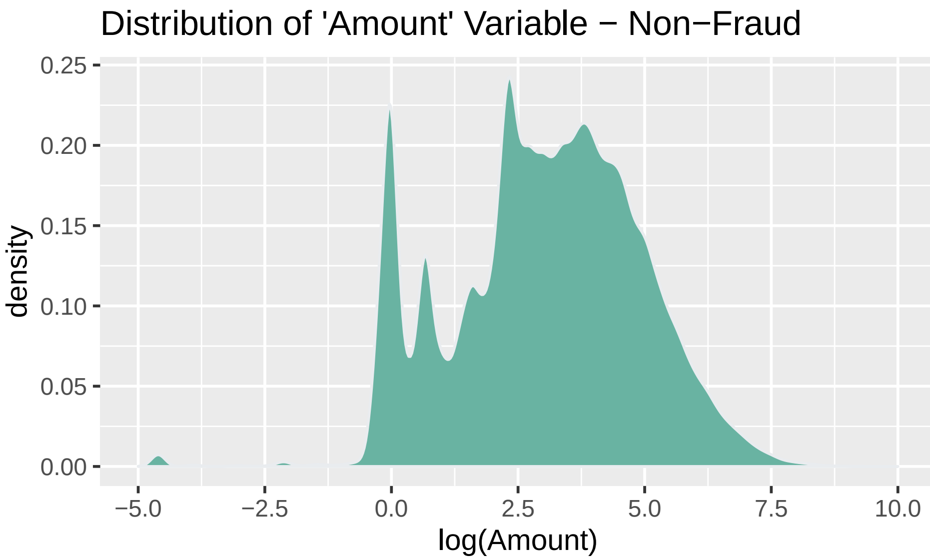
1. Non-fraud credit card transactions started to pick up from 08:00AM, maintained traction, and fell off at roughly 22:00PM
2. This would be in line with how a normal person would be using his/her credit card; starting from breakfast hours, throughout the day, and tapering off when the day ends
3. Fraudulent credit card transactions had more activity during 00:00 midnight to 08:00AM
4. This could be because credit card transactions made during these hours, when the real owners are presumably asleep, are less likely to be found out via real-time bank alerts etc.
5. This could also suggest fraudulent credit card transactions are made in a different time zone

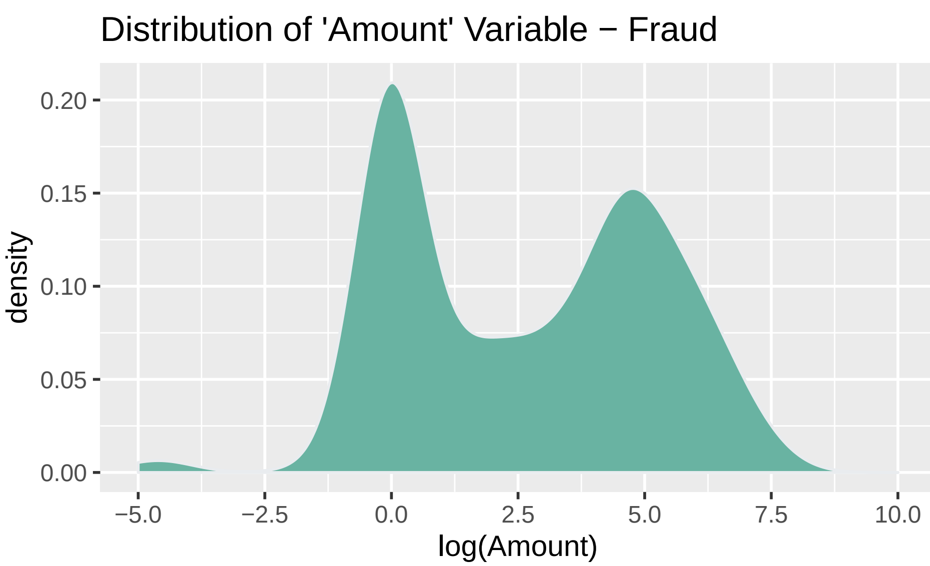
Part 2.4: Inspect 'Amount' variable





- 'Amount' variable cannot be seen properly without log transformation
- Hence the log transformed 'Amount' variable will be used in the charts below





Insights from Part 2.4

1. The summary table shows the min. for 'Amount' variable is 0.00, which seems counter-intuitive to a credit card transaction
2. Potentially, a 0.00 amount could be used to verify if the credit card is a valid one
3. Fraudulent transactions have amounts smoothened out with less variability in the distribution
4. This may suggest certain 'favored' amounts that credit card fraudsters charge to the credit cards
5. This is unlike non-fraudulent transactions, where credit card transactions can be for a wide range of amounts since transactions for goods and services are unlimited to certain amounts

STEP 3: DATA PREPROCESSING

Part 3.1: Rescale variables to be in the range 0 to 1

- Variables with higher values may dominate computations and skew the model performance
- Hence, normalize data to work with different variables that are in different scales

```
##                min          max
## Time      0.000000 1.727920e+05
## V1      -56.407510 2.454930e+00
## V2      -72.715728 2.205773e+01
## V3      -48.325589 9.382558e+00
## V4       -5.683171 1.687534e+01
## V5     -113.743307 3.480167e+01
## V6      -26.160506 7.330163e+01
## V7      -43.557242 1.205895e+02
## V8      -73.216718 2.000721e+01
## V9      -13.434066 1.559499e+01
## V10     -24.588262 2.374514e+01
## V11      -4.797473 1.201891e+01
## V12     -18.683715 7.848392e+00
## V13      -5.791881 7.126883e+00
## V14     -19.214325 1.052677e+01
## V15      -4.498945 8.877742e+00
## V16     -14.129855 1.731511e+01
## V17     -25.162799 9.253526e+00
## V18      -9.498746 5.041069e+00
## V19      -7.213527 5.591971e+00
## V20     -54.497720 3.942090e+01
## V21     -34.830382 2.720284e+01
## V22     -10.933144 1.050309e+01
## V23     -44.807735 2.252841e+01
## V24      -2.836627 4.584549e+00
## V25     -10.295397 7.519589e+00
## V26      -2.604551 3.517346e+00
## V27     -22.565679 3.161220e+01
## V28     -15.430084 3.384781e+01
## Amount    0.000000 2.569116e+04
## hour      0.000000 2.399944e+01
```

```
##        min_after_rescaling max_after_rescaling
## Time                     0                   1
## V1                       0                   1
## V2                       0                   1
## V3                       0                   1
## V4                       0                   1
## V5                       0                   1
## V6                       0                   1
## V7                       0                   1
## V8                       0                   1
## V9                       0                   1
## V10                      0                   1
## V11                      0                   1
## V12                      0                   1
```

```
## V13                        0                    1
## V14                        0                    1
## V15                        0                    1
## V16                        0                    1
## V17                        0                    1
## V18                        0                    1
## V19                        0                    1
## V20                        0                    1
## V21                        0                    1
## V22                        0                    1
## V23                        0                    1
## V24                        0                    1
## V25                        0                    1
## V26                        0                    1
## V27                        0                    1
## V28                        0                    1
## Amount                     0                    1
## hour                       0                    1
```

STEP 4: SPLIT DATASET INTO TRAIN & TEST SET

Set seed

```r
set.seed(1)
```

Part 4.1: Baseline dataset

Train data: 70%; Test data: 30%

```r
index_train <- createDataPartition(data$Class, p=0.7, list=FALSE)

baseline_train <- data[index_train, ]
test <- data[-index_train, ]

baseline_train$Class <- factor(baseline_train$Class)
```

Part 4.2: MWMOTE adjusted dataset

- MWMOTE: Majority Weighted Minority Oversampling Technique
- Oversampling helps to balance class distribution by duplicating minority class instances

Generate more noisy instances out of dataset

```
mwmote_train_gen <- mwmote(as.data.frame(baseline_train), numInstances=100000)
table(mwmote_train_gen$Class)
```

```
##
##    one   zero
## 100000      0
```

Bind the train dataset and the newly generated instances

```
mwmote_train <- rbind(baseline_train, mwmote_train_gen)
```

Compare 'Class' variable in the baseline dataset and MWMOTE adjusted dataset

```
table(baseline_train$Class)
```

```
##
##    one   zero
##    345 199021
```

```
table(mwmote_train$Class)
```

```
##
##    one   zero
## 100345 199021
```

```
mwmote_train$Class <- factor(mwmote_train$Class)
```

Insights from Part 4.2

1. MWMOTE adjusted dataset is a more balanced dataset

Part 4.3: SMOTE adjusted dataset

- Undersample instances of the majority class so classifiers are not biased toward one class

```
set.seed(1)
smote_train_gen1 <- SMOTE(X = baseline_train[, -1], target = baseline_train$Class, dup_size = 4)
smote_train_gen2 <- smote_train_gen1$data %>% rename(Class = class)
```

Undersample dataset until majority class size matches

```
smote_train_gen3 <- ovun.sample(Class ~ .,
                        data = smote_train_gen2,
                        method = "under",
                        N = 2 * sum(smote_train_gen2$Class == "one"))

smote_train <- smote_train_gen3$data
```

Compare 'Class' variable in the baseline dataset and SMOTE adjusted dataset

```
table(baseline_train$Class)
```

```
##
##    one   zero
##    345 199021
```

```
table(smote_train$Class)
```

```
##
##  one zero
## 1725 1725
```

```
smote_train$Class <- factor(smote_train$Class)
```

Insights from Part 4.3

1. SMOTE adjusted dataset is a more balanced dataset

STEP 5: MODEL SELECTION & TRAINING

- Evaluation Metric: Area Under the Precision-Recall Curve (AUPRC)

XGBoost: xgbTree Method

- Build a series of trees where each tree is trained to attempt to correct the mistakes of the previous tree in the series
- To create a model that makes fewer and fewer mistakes as more trees are added
- Making predictions with gradient boosted tree models is faster

Part 5.1: XGBoost model for baseline dataset

```
##    nrounds max_depth eta gamma colsample_bytree min_child_weight subsample
## 32     100         2 0.3     0              0.8                1      0.75
```

```
## [1] "Area Under the Precision-Recall Curve: 0.873"
```

Add baseline model results to a table for later comparison

Model	AUPRC
Baseline XGBoost	0.873

Part 5.2: XGBoost model for MWMOTE adjusted dataset

```
##    nrounds max_depth eta gamma colsample_bytree min_child_weight subsample
## 54     150         3 0.3     0              0.8                1         1
```

```
## [1] "Area Under the Precision-Recall Curve: 0.86"
```

Add baseline model results to a table for later comparison

Model	AUPRC
Baseline XGBoost	0.873
MWMOTE Adjusted XGBoost	0.860

Part 5.3: XGBoost model for SMOTE adjusted dataset

```
##    nrounds max_depth eta gamma colsample_bytree min_child_weight subsample
## 51     150         3 0.3     0              0.8                1      0.75
```

```
## [1] "Area Under the Precision-Recall Curve: 0.722"
```

Add baseline model results to a table for later comparison

Model	AUPRC
Baseline XGBoost	0.873
MWMOTE Adjusted XGBoost	0.860
SMOTE Adjusted XGBoost	0.722

RESULTS

Modeling Results & Model Performance

To recap, the goal of this project is to identify fraudulent credit card transactions. As the dataset is highly imbalanced, the accuracy will be measured using the Area Under the Precision-Recall Curve (AUPRC). The modeling results and model performance shows that the baseline training dataset delivered better model performance. In summary, the highest AUPRC was **0.873**, achieved using **XGBoost** with **xgbTree** method.

This reflects the challenges in dealing with a highly imbalanced dataset. An examination of the XGBoost model results shows that the *max_depth* (which controls the depth of the tree) using the baseline train dataset was **2**, while using the MWMOTE adjusted and SMOTE adjusted train datasets returned a *max_depth* of **3**. This could potentially explain why the baseline trained model scored the highest AUPRC. It is possible that both models trained with the adjusted datasets were overfitting.

CONCLUSION

Brief Summary

In summary, this report explained the steps in creating a machine learning algorithm to detect fraudulent credit card transactions from the Credit Card Fraud Detection dataset (from Kaggle). This report also dealt with a highly imbalanced dataset and included trying out various preprocessing steps for such datasets. The boosting method from the **XGBoost** package is used for this classification problem, with Area under the Precision-Recall curve of **0.873**.

Limitations and Future Work

For the purpose of this project, the Credit Card Fraud Detection dataset provided by Kaggle had most of the variables masked via PCA. As such, while the algorithms may accurately detect a fraudulent credit card transaction, it does not provide a rational explanation as to why a transaction was categorized as fraudulent. For future work, with unmasked variables, we may use the machine learning algorithm to better understand which are the variables that would lead to a transaction being classified as fraudulent, and may further work towards this to reduce credit card frauds.

Github

The following link to the GitHub repository contains the reports in PDF format, Rmd format and R script: https://github.com/Nicole-Yong/Credit-Card-Fraud-Analysis