Capital One Data Science Challenge

2/8/2021

Contents

Dataset Preparation

Downloading and Unzipping the dataset.

```
url <- "https://github.com/CapitalOneRecruiting/DS/raw/master/transactions.zip"
curl::curl_download(url,destfile = "data.zip")
unzip("data.zip")
```

Loading into memory as a dataframe

```
library(jsonlite)
library(tidyverse)
library(tidylog)
library(vroom)
```

```
raw_data <- vroom_lines("transactions.txt")
df <- map_dfr(raw_data,parse_json)
```

Summary Statistics for all atributes

```
library(summarytools)

dfSummary(df, plain.ascii = FALSE, style = "grid", graph.col = FALSE,
          valid.col = FALSE, tmp.img.dir = "/tmp")
```

```
### Data Frame Summary
#### df
**Dimensions:** 786363 x 29
**Duplicates:** 0

+----+---------------------------+----------------------------------+--------------------------+---------+
| No | Variable                  | Stats / Values                   | Freqs (% of Valid)       | Missing |
+====+===========================+==================================+==========================+=========+
| 1  | accountNumber\            | 1\. 380680241\                   | 32850 ( 4.2%)\           | 0\      |
|    | [character]               | 2\. 882815134\                   | 13189 ( 1.7%)\           | (0.0%)  |
|    |                           | 3\. 570884863\                   | 10867 ( 1.4%)\           |         |
|    |                           | 4\. 246251253\                   | 10172 ( 1.3%)\           |         |
|    |                           | 5\. 369308035\                   | 7229 ( 0.9%)\            |         |
|    |                           | 6\. 724518977\                   | 6283 ( 0.8%)\            |         |
|    |                           | 7\. 894938833\                   | 6101 ( 0.8%)\            |         |
|    |                           | 8\. 419709514\                   | 5930 ( 0.8%)\            |         |
|    |                           | 9\. 832842201\                   | 5850 ( 0.7%)\            |         |
|    |                           | 10\. 208319653\                  | 5235 ( 0.7%)\            |         |
|    |                           | [ 4990 others ]                  | 682657 (86.8%)           |         |
+----+---------------------------+----------------------------------+--------------------------+---------+
| 2  | customerId\               | 1\. 380680241\                   | 32850 ( 4.2%)\           | 0\      |
|    | [character]               | 2\. 882815134\                   | 13189 ( 1.7%)\           | (0.0%)  |
|    |                           | 3\. 570884863\                   | 10867 ( 1.4%)\           |         |
|    |                           | 4\. 246251253\                   | 10172 ( 1.3%)\           |         |
|    |                           | 5\. 369308035\                   | 7229 ( 0.9%)\            |         |
```

		6\. 724518977\	6283 (0.8%)\		
		7\. 894938833\	6101 (0.8%)\		
		8\. 419709514\	5930 (0.8%)\		
		9\. 832842201\	5850 (0.7%)\		
		10\. 208319653\	5235 (0.7%)\		
		[4990 others]	682657 (86.8%)		
3	creditLimit\ [numeric]	Mean (sd) : 10759.5 (11636.2)\ min < med < max:\ 250 < 7500 < 50000\ IQR (CV) : 10000 (1.1)	250 : 34025 (4.3%)\ 500 : 27097 (3.4%)\ 1000 : 36430 (4.6%)\ 2500 : 75429 (9.6%)\ 5000 : 201863 (25.7%)\ 7500 : 97913 (12.5%)\ 10000 : 56889 (7.2%)\ 15000 : 139307 (17.7%)\ 20000 : 68629 (8.7%)\ 50000 : 48781 (6.2%)	0\ (0.0%)	
4	availableMoney\ [numeric]	Mean (sd) : 6250.7 (8880.8)\ min < med < max:\ -1005.6 < 3184.9 < 50000\ IQR (CV) : 6422.6 (1.4)	521694 distinct values	0\ (0.0%)	
5	transactionDateTime\ [character]	1\. 2016-05-28T14:24:41\ 2\. 2016-12-25T14:04:15\ 3\. 2016-01-26T08:54:30\ 4\. 2016-01-26T23:43:32\ 5\. 2016-02-13T11:03:34\ 6\. 2016-02-21T23:29:36\ 7\. 2016-02-25T17:00:11\ 8\. 2016-02-26T19:57:50\ 9\. 2016-03-19T02:54:51\ 10\. 2016-03-20T05:22:32\ [776627 others]	4 (0.0%)\ 4 (0.0%)\ 3 (0.0%)\ 3 (0.0%)\ 3 (0.0%)\ 3 (0.0%)\ 3 (0.0%)\ 3 (0.0%)\ 3 (0.0%)\ 3 (0.0%)\ 786331 (100.0%)	0\ (0.0%)	
6	transactionAmount\ [numeric]	Mean (sd) : 137 (147.7)\ min < med < max:\ 0 < 87.9 < 2011.5\ IQR (CV) : 157.8 (1.1)	66038 distinct values	0\ (0.0%)	
7	merchantName\ [character]	1\. Uber\ 2\. Lyft\ 3\. oldnavy.com\ 4\. staples.com\ 5\. alibaba.com\ 6\. apple.com\ 7\. walmart.com\ 8\. cheapfast.com\ 9\. ebay.com\ 10\. target.com\ [2480 others]	25613 (3.3%)\ 25523 (3.2%)\ 16992 (2.2%)\ 16980 (2.2%)\ 16959 (2.2%)\ 16898 (2.1%)\ 16873 (2.1%)\ 16858 (2.1%)\ 16842 (2.1%)\ 16813 (2.1%)\ 600012 (76.3%)	0\ (0.0%)	
8	acqCountry\ [character]	1\. (Empty string)\ 2\. CAN\	4562 (0.6%)\ 2424 (0.3%)\	0\ (0.0%)	

		3\. MEX\	3130 (0.4%)\		
		4\. PR\	1538 (0.2%)\		
		5\. US	774709 (98.5%)		
9	merchantCountryCode\ [character]	1\. (Empty string)\	724 (0.1%)\	0\ (0.0%)	
		2\. CAN\	2426 (0.3%)\		
		3\. MEX\	3143 (0.4%)\		
		4\. PR\	1559 (0.2%)\		
		5\. US	778511 (99.0%)		
10	posEntryMode\ [character]	1\. (Empty string)\	4054 (0.5%)\	0\ (0.0%)	
		2\. 02\	195934 (24.9%)\		
		3\. 05\	315035 (40.1%)\		
		4\. 09\	236481 (30.1%)\		
		5\. 80\	15283 (1.9%)\		
		6\. 90	19576 (2.5%)		
11	posConditionCode\ [character]	1\. (Empty string)\	409 (0.1%)\	0\ (0.0%)	
		2\. 01\	628787 (80.0%)\		
		3\. 08\	149634 (19.0%)\		
		4\. 99	7533 (1.0%)		
12	merchantCategoryCode\ [character]	1\. online_retail\	202156 (25.7%)\	0\ (0.0%)	
		2\. fastfood\	112138 (14.3%)\		
		3\. entertainment\	80098 (10.2%)\		
		4\. food\	75490 (9.6%)\		
		5\. online_gifts\	66238 (8.4%)\		
		6\. rideshare\	51136 (6.5%)\		
		7\. hotels\	34097 (4.3%)\		
		8\. fuel\	23910 (3.0%)\		
		9\. subscriptions\	22901 (2.9%)\		
		10\. auto\	21651 (2.8%)\		
		[9 others]	96548 (12.3%)		
13	currentExpDate\ [character]	1\. 03/2029\	5103 (0.6%)\	0\ (0.0%)	
		2\. 08/2024\	5087 (0.6%)\		
		3\. 10/2023\	5075 (0.6%)\		
		4\. 05/2027\	5063 (0.6%)\		
		5\. 01/2021\	5041 (0.6%)\		
		6\. 10/2032\	5039 (0.6%)\		
		7\. 08/2030\	5029 (0.6%)\		
		8\. 08/2022\	5026 (0.6%)\		
		9\. 07/2031\	5022 (0.6%)\		
		10\. 05/2026\	5021 (0.6%)\		
		[155 others]	735857 (93.6%)		
14	accountOpenDate\ [character]	1\. 2014-06-21\	33623 (4.3%)\	0\ (0.0%)	
		2\. 2014-09-30\	13335 (1.7%)\		
		3\. 2014-05-22\	11353 (1.4%)\		
		4\. 2012-10-09\	10867 (1.4%)\		
		5\. 2014-10-02\	10653 (1.4%)\		
		6\. 2014-11-05\	6693 (0.9%)\		
		7\. 2014-12-04\	6584 (0.8%)\		
		8\. 2015-03-01\	5962 (0.8%)\		

		9\. 2015-02-21\	5881 (0.7%)\		
		10\. 2015-08-14\	5503 (0.7%)\		
		[1810 others]	675909 (86.0%)		
15	dateOfLastAddressChange\ [character]	1\. 2016-03-15\	3819 (0.5%)\	0\ (0.0%)	
		2\. 2016-01-06\	3740 (0.5%)\		
		3\. 2016-01-04\	3558 (0.5%)\		
		4\. 2016-06-08\	3355 (0.4%)\		
		5\. 2016-04-04\	3194 (0.4%)\		
		6\. 2016-02-22\	3102 (0.4%)\		
		7\. 2016-03-23\	3091 (0.4%)\		
		8\. 2016-04-07\	3036 (0.4%)\		
		9\. 2016-05-19\	2969 (0.4%)\		
		10\. 2016-01-01\	2925 (0.4%)\		
		[2174 others]	753574 (95.8%)		
16	cardCVV\ [character]	1\. 869\	33749 (4.3%)\	0\ (0.0%)	
		2\. 289\	15509 (2.0%)\		
		3\. 640\	10804 (1.4%)\		
		4\. 455\	10279 (1.3%)\		
		5\. 959\	7024 (0.9%)\		
		6\. 917\	6503 (0.8%)\		
		7\. 548\	5487 (0.7%)\		
		8\. 586\	5245 (0.7%)\		
		9\. 296\	5125 (0.7%)\		
		10\. 494\	5038 (0.6%)\		
		[889 others]	681600 (86.7%)		
17	enteredCVV\ [character]	1\. 869\	33424 (4.3%)\	0\ (0.0%)	
		2\. 289\	15401 (2.0%)\		
		3\. 640\	10731 (1.4%)\		
		4\. 455\	10176 (1.3%)\		
		5\. 959\	6963 (0.9%)\		
		6\. 917\	6440 (0.8%)\		
		7\. 548\	5438 (0.7%)\		
		8\. 586\	5202 (0.7%)\		
		9\. 296\	5081 (0.6%)\		
		10\. 494\	5005 (0.6%)\		
		[966 others]	682502 (86.8%)		
18	cardLast4Digits\ [character]	1\. 593\	32946 (4.2%)\	0\ (0.0%)	
		2\. 2194\	10867 (1.4%)\		
		3\. 6002\	10172 (1.3%)\		
		4\. 6580\	6747 (0.9%)\		
		5\. 8502\	6553 (0.8%)\		
		6\. 2178\	5930 (0.8%)\		
		7\. 0000\	5492 (0.7%)\		
		8\. 7629\	5235 (0.7%)\		
		9\. 2701\	4826 (0.6%)\		
		10\. 2864\	4533 (0.6%)\		
		[5236 others]	693062 (88.1%)		
19	transactionType\ [character]	1\. (Empty string)\	698 (0.1%)\	0\ (0.0%)	
		2\. ADDRESS_VERIFICATION\	20169 (2.6%)\		

		3\. PURCHASE\	745193 (94.8%)\		
		4\. REVERSAL	20303 (2.6%)		
20	echoBuffer\ [character]	All empty strings		0\ (0.0%)	
21	currentBalance\ [numeric]	Mean (sd) : 4508.7 (6457.4)\ min < med < max:\ 0 < 2451.8 < 47498.8\ IQR (CV) : 4601.2 (1.4)	487318 distinct values	0\ (0.0%)	
22	merchantCity\ [character]	All empty strings		0\ (0.0%)	
23	merchantState\ [character]	All empty strings		0\ (0.0%)	
24	merchantZip\ [character]	All empty strings		0\ (0.0%)	
25	cardPresent\ [logical]	1\. FALSE\ 2\. TRUE	433495 (55.1%)\ 352868 (44.9%)	0\ (0.0%)	
26	posOnPremises\ [character]	All empty strings		0\ (0.0%)	
27	recurringAuthInd\ [character]	All empty strings		0\ (0.0%)	
28	expirationDateKeyInMatch\ [logical]	1\. FALSE\ 2\. TRUE	785320 (99.9%)\ 1043 (0.1%)	0\ (0.0%)	
29	isFraud\ [logical]	1\. FALSE\ 2\. TRUE	773946 (98.4%)\ 12417 (1.6%)	0\ (0.0%)	

Data Cleaning

Converting all empty string to NA in column of type string

```
df <-
  df %>%
  mutate(across(where(is_character), na_if,""))
```

Dropping echoBuffer, merchantCity, merchantState, merchantZip, posOnPremises and recurringAuthInd since these have all observations missing.

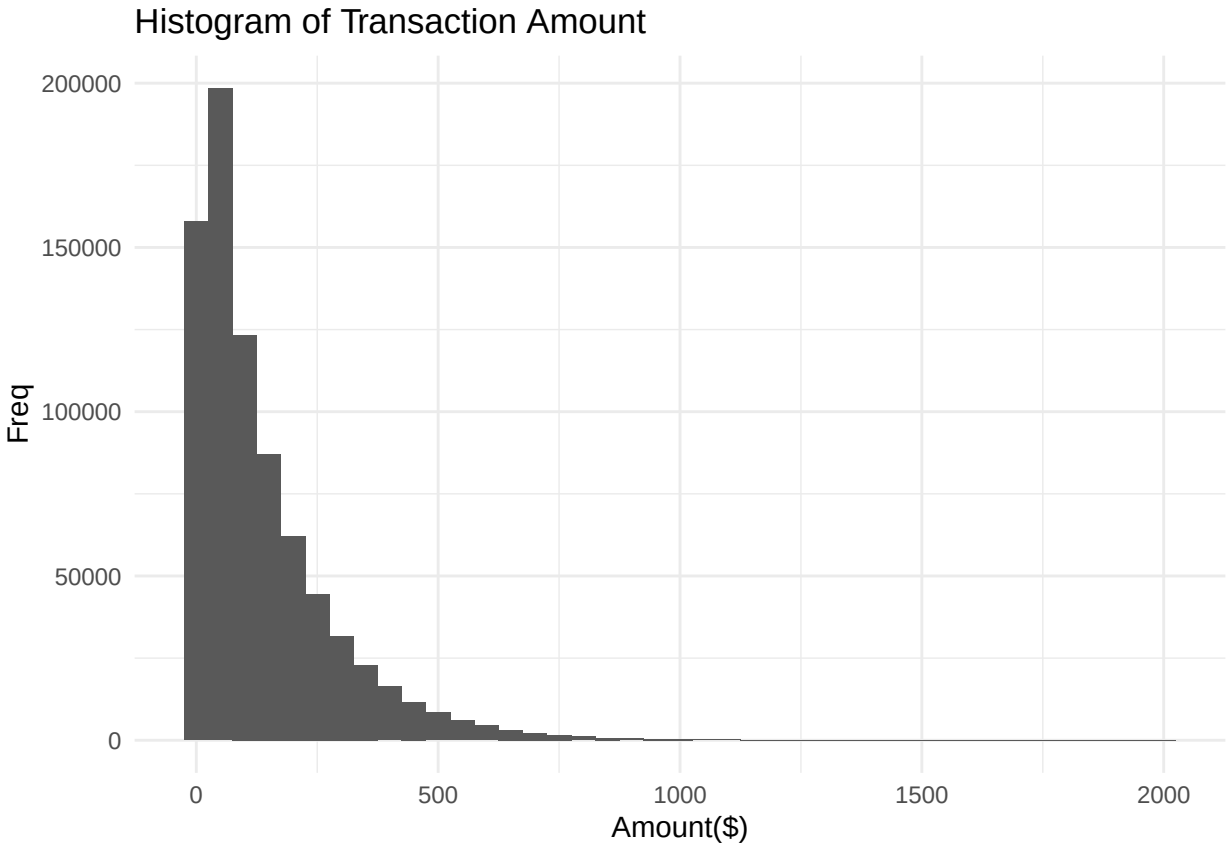
```
df <-
df %>%
  select(-echoBuffer, -merchantCity, -merchantState, -merchantZip, -posOnPremises, -recurringAuthInd)
```

Changing date column to date data type.

```
library(janitor)
library(lubridate)

df <-
  df %>%
  mutate(transactionDateTime = ymd_hms(transactionDateTime),
         currentExpDate = my(currentExpDate),
         accountOpenDate = ymd(accountOpenDate),
         dateOfLastAddressChange = ymd(dateOfLastAddressChange))
```

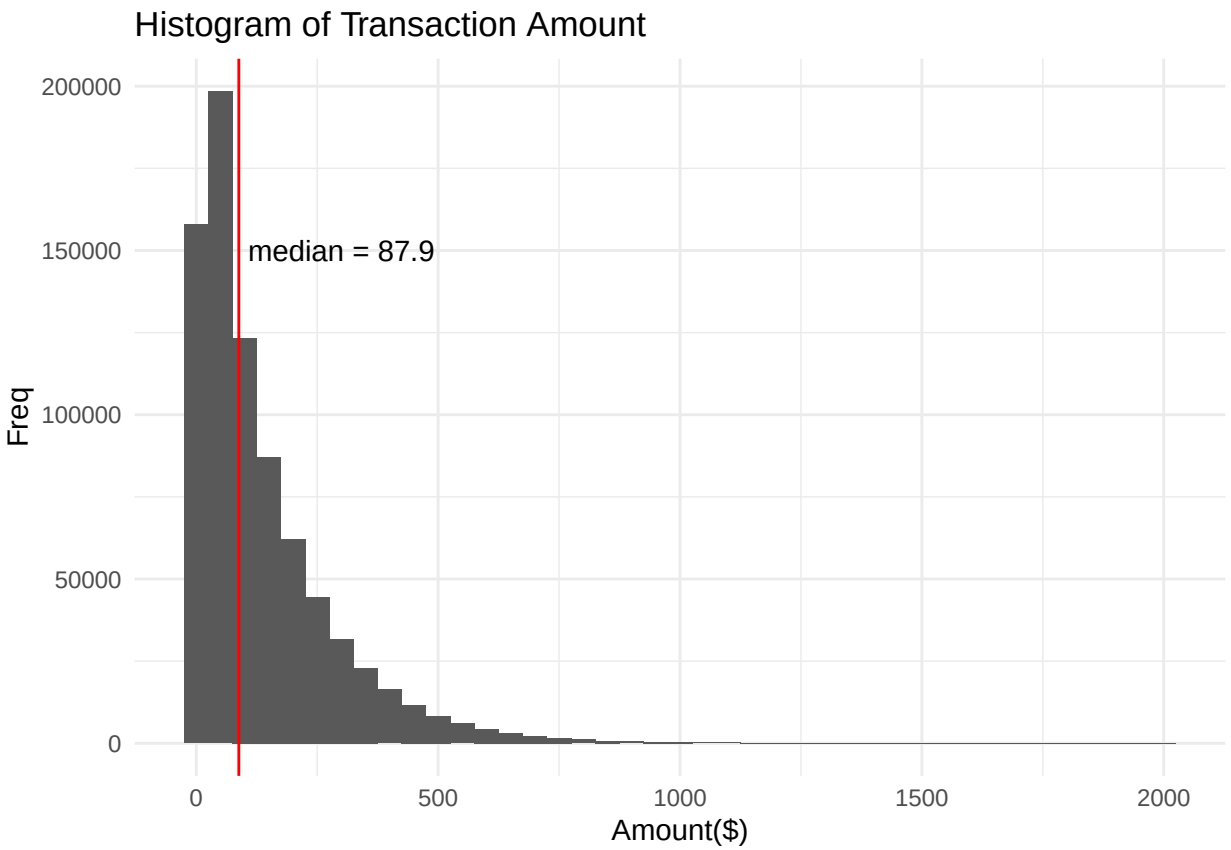
Visualizing distribution of Transaction Amount

```
df %>%
  ggplot(aes(transactionAmount)) +
  geom_histogram(binwidth = 50)+
  labs(
    title = "Histogram of Transaction Amount",
    x = "Amount($)",
    y = "Freq"
  )+
  theme_minimal()
```



Here most of the transaction lies below $100 and the distribution is positively skewed.

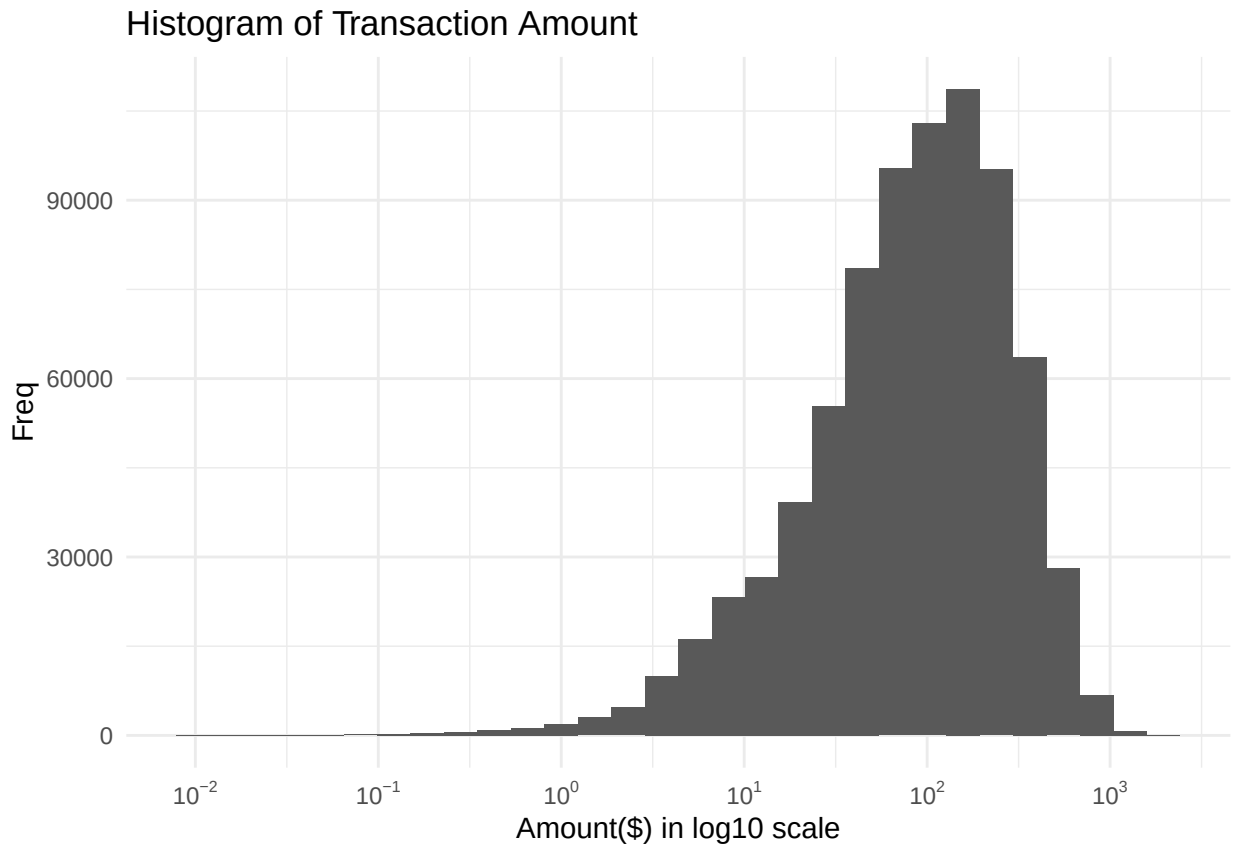
```
library(glue)
df %>%
  ggplot(aes(transactionAmount)) +
  geom_histogram(binwidth = 50)+
  labs(
    title = "Histogram of Transaction Amount",
    x = "Amount($)",
    y = "Freq"
  )+
  geom_vline(xintercept = median(df$transactionAmount), color="red")+
  annotate("text",x=300,y=150000,label=glue("median = {median(df$transactionAmount)}"))+
  theme_minimal()
```



This distribution looks like a log normal distribution.

```
library(scales)
df %>%
  ggplot(aes(transactionAmount)) +
  geom_histogram()+
  labs(
    title = "Histogram of Transaction Amount",
    x = "Amount($) in log10 scale",
    y = "Freq"
  )+
  scale_x_log10(breaks = trans_breaks("log10", function(x) 10^x),
```

```
              labels = trans_format("log10", math_format(10^.x))) +
  theme_minimal()
```



Data Wrangling

Reversed Transaction

Looking into Purchased and Reversed pairs of transaction(ordered by datetime) per customer.
Here I am grouping the dataset by customer id and checking if transaction type matches with its lagged version.

```
df_rev <-
df %>%
  filter(transactionType %in% c("PURCHASE","REVERSAL")) %>%
  group_by(customerId) %>%
  arrange(transactionDateTime) %>%
  group_map(~ (mutate(.x,transactionTypeLagged = lead(transactionType,default = "PURCHASE")) %>%
               filter(transactionType != transactionTypeLagged))) %>%
  bind_rows() %>%
  ungroup()

df_rev %>%
  select(accountNumber,transactionDateTime,transactionAmount, transactionType) %>%
```

```
  head(10) %>%
  knitr::kable()
```

accountNumber	transactionDateTime	transactionAmount	transactionType
100088067	2016-11-20 07:57:05	22.32	PURCHASE
100088067	2016-11-20 08:00:04	22.32	REVERSAL
100328049	2016-01-15 20:34:35	43.74	PURCHASE
100328049	2016-01-15 20:36:18	43.74	REVERSAL
100328049	2016-03-24 22:57:15	284.97	PURCHASE
100328049	2016-03-26 17:35:09	284.97	REVERSAL
100737756	2016-06-05 20:44:58	130.49	PURCHASE
100737756	2016-06-06 22:56:52	93.67	REVERSAL
100737756	2016-08-24 05:27:01	57.94	PURCHASE
100737756	2016-08-24 20:21:19	501.29	REVERSAL

Not all Purchased and Reversed pairs are actually Reversed Transaction. So,
For a transaction to be reversed, a normal purchase should be followed by a reversed and both should have same transaction amount.

```
df_rev_eq <-
df_rev %>%
  mutate(index = ceiling((1: n())/2)) %>%
  group_by(index) %>%
  mutate(avg_amount = sum(transactionAmount)/2) %>%
  ungroup() %>%
  filter(transactionAmount == avg_amount) %>%
  select(-index, -avg_amount, -transactionTypeLagged)

df_rev_eq %>%
  select(accountNumber,transactionDateTime,transactionAmount, transactionType) %>%
  head(10) %>%
  knitr::kable()
```

accountNumber	transactionDateTime	transactionAmount	transactionType
100088067	2016-11-20 07:57:05	22.32	PURCHASE
100088067	2016-11-20 08:00:04	22.32	REVERSAL
100328049	2016-01-15 20:34:35	43.74	PURCHASE
100328049	2016-01-15 20:36:18	43.74	REVERSAL
100328049	2016-03-24 22:57:15	284.97	PURCHASE
100328049	2016-03-26 17:35:09	284.97	REVERSAL
101376441	2016-07-05 15:03:41	96.63	PURCHASE
101376441	2016-07-12 00:44:32	96.63	REVERSAL
101596991	2016-12-30 19:49:59	294.01	PURCHASE
101596991	2016-12-30 19:50:06	294.01	REVERSAL

So total reversal amount and count is:

```
df_rev_eq %>%
  filter(transactionType == "REVERSAL") %>%
```

```
  summarise(TotalReversalAmount = sum(transactionAmount),
            TotalReversalCount = n()) %>%
  knitr::kable()
```

TotalReversalAmount	TotalReversalCount
649501	4339

Removing all Purchased and Reversed pairs

```
df <-
df %>%
  anti_join(df_rev_eq)
```

Multi-swipe Transaction

A transaction is multi-swiped if same transaction occur multiple times but in short duration. I am assuming short duration to be 10 minutes.
Here I am grouping the dataset by customer ID and again grouping it by 10 minute duration. I am checking for similar transaction in the group.

```
df_multi <-
df %>%
  filter(transactionType == "PURCHASE") %>%
  group_by(customerId) %>%
  mutate(date_index = floor_date(transactionDateTime,unit = "10minutes")) %>%
  group_by(date_index) %>%
  filter(duplicated(transactionAmount) | duplicated(transactionAmount,fromLast = T),
         duplicated(merchantName) | duplicated(merchantName,fromLast = T)) %>%
  ungroup()

df_multi %>%
  arrange(customerId,transactionDateTime) %>%
  select(accountNumber,transactionDateTime,transactionAmount, merchantName) %>%
  head(15) %>%
  knitr::kable()
```

accountNumber	transactionDateTime	transactionAmount	merchantName
100088067	2016-10-16 18:01:00	411.35	Fresh Flowers
100088067	2016-10-16 18:01:02	411.35	Fresh Flowers
100737756	2016-01-18 01:55:24	693.50	Franks Deli
100737756	2016-01-18 01:55:28	693.50	Franks Deli
100737756	2016-01-18 01:58:26	693.50	Franks Deli
100737756	2016-07-02 12:05:04	211.22	South Steakhouse #73819
100737756	2016-07-02 12:07:00	211.22	South Steakhouse #73819
100737756	2016-07-10 14:31:07	43.25	34th BBQ #166379
100737756	2016-07-10 14:32:06	43.25	34th BBQ #166379
101132326	2016-08-24 02:09:08	188.86	Regal Cinemas #05791
101132326	2016-08-24 02:09:44	188.86	Regal Cinemas #05791
101380713	2016-03-13 12:01:27	29.28	sears.com

accountNumber	transactionDateTime	transactionAmount	merchantName
101380713	2016-03-13 12:03:55	29.28	sears.com
101380713	2016-07-23 06:53:44	33.74	amazon.com
101380713	2016-07-23 06:56:15	33.74	amazon.com

Total number of transactions and total dollar amount for the multi-swipe transactions

```
# Considering First transaction to be normal

df_multi_normal <-
  df_multi %>%
  group_by(customerId,date_index) %>%
  distinct(transactionAmount,merchantName,.keep_all = T) %>%
  ungroup()

df_multi %>%
  anti_join(df_multi_normal) %>%
  summarise(totalMultiTransacAmt = sum(transactionAmount),
            totalMultiTransacCount = n()) %>%
  knitr::kable()
```

totalMultiTransacAmt	totalMultiTransacCount
928906.1	6226

Removing all Multi-swipe transaction

```
df <-
df %>%
  anti_join(df_multi %>%
              anti_join(df_multi_normal))
```

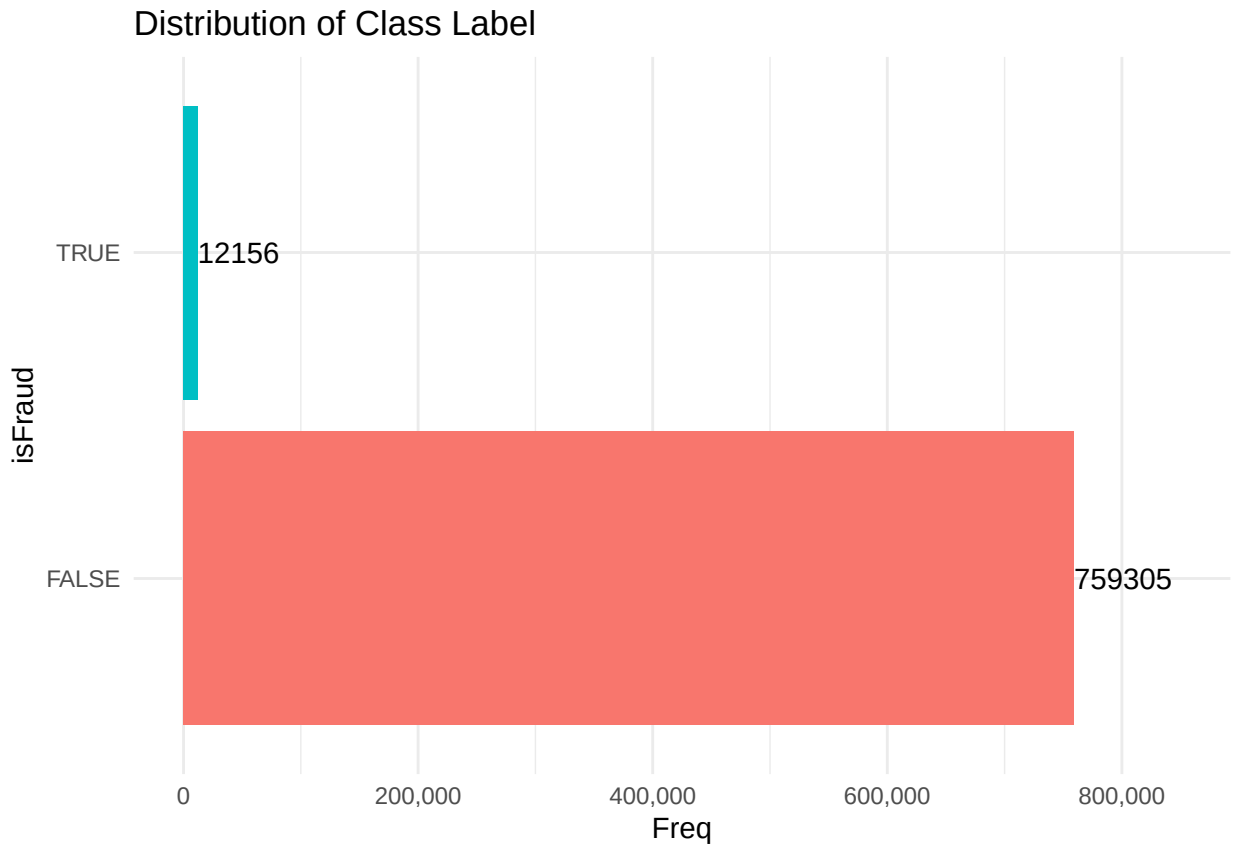
Modelling

Balancing the dataset

Our dataset is highly unbalanced.

```
df %>%
  count(isFraud) %>%
  ggplot(aes(n,isFraud,fill=isFraud)) +
  geom_col()+
  guides(fill = FALSE)+
  labs(
    title = "Distribution of Class Label",
    x = "Freq"
  )+
  geom_text(aes(label = n),hjust=0)+
  scale_x_continuous(labels = scales::comma ,
```

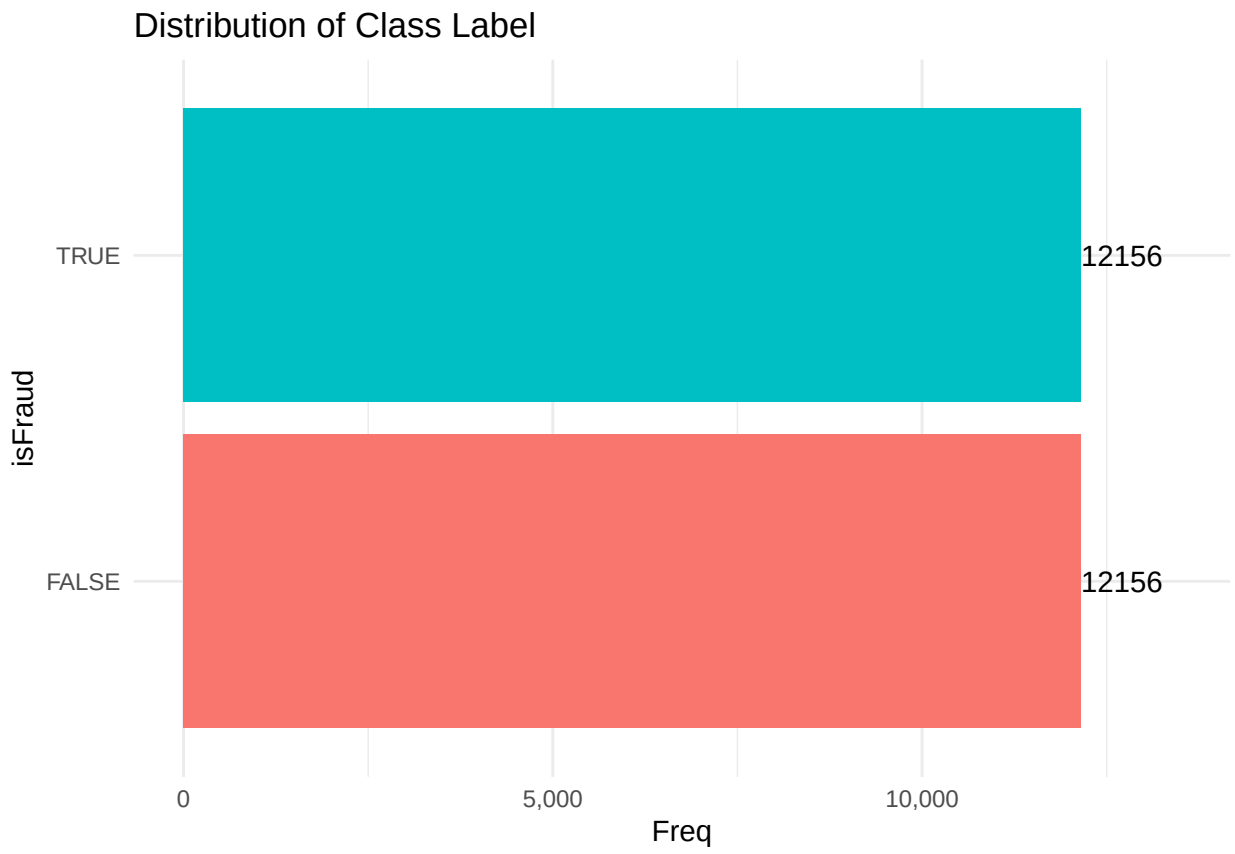
```
                    limits = c(0,850000))+
  theme_minimal()
```



We have to sample our dataset to make it balanced. Here I am undersampling 12156 transaction from both Fraud and not Fraud transactions.

```
df_sample <-
  df %>%
  group_by(isFraud) %>%
  sample_n(12156) %>%
  ungroup()
```

```
df_sample %>%
  count(isFraud) %>%
  ggplot(aes(n,isFraud,fill=isFraud)) +
  geom_col()+
  guides(fill = FALSE)+
  labs(
    title = "Distribution of Class Label",
    x = "Freq"
  )+
  geom_text(aes(label = n),hjust=0)+
  scale_x_continuous(labels = scales::comma ,
                     limits = c(0,13500))+
  theme_minimal()
```



Feature Engineering

Feature Selection Some attributes that obviously don't have any predictive power
1. accountNumber
2. customerId
3. cardLast4Digits

```
df_sample <-
  df_sample %>%
  select(-accountNumber, -customerId, -cardLast4Digits)
```

Merchant Too many merchant and their frequencies are low.

```
df_sample %>%
  count(merchantName)
```

```
## # A tibble: 1,900 x 2
##    merchantName                 n
##  * <chr>                    <int>
##  1 1st BBQ                     34
##  2 1st Deli                    13
##  3 1st Pub                     35
##  4 1st Restaurant              26
##  5 1st Sandwitch Bar #119707   28
```

```
##  6 1st Sandwitch Bar #396252    19
##  7 1st Sandwitch Bar #758805    21
##  8 1st Sandwitch Bar #772439    22
##  9 1st Sandwitch Bar #801388    22
## 10 1st Sandwitch Bar #941288    25
## # ... with 1,890 more rows
```

Count for merchant category.

```
df_sample %>%
  count(merchantCategoryCode) %>%
  knitr::kable()
```

merchantCategoryCode	n
airline	754
auto	611
cable/phone	22
entertainment	2118
fastfood	2754
food	2203
food_delivery	109
fuel	397
furniture	205
gym	35
health	351
hotels	762
mobileapps	241
online_gifts	2586
online_retail	8007
online_subscriptions	162
personal care	401
rideshare	2019
subscriptions	575

Lets keep top 6 merchant category and lump remaining to other.

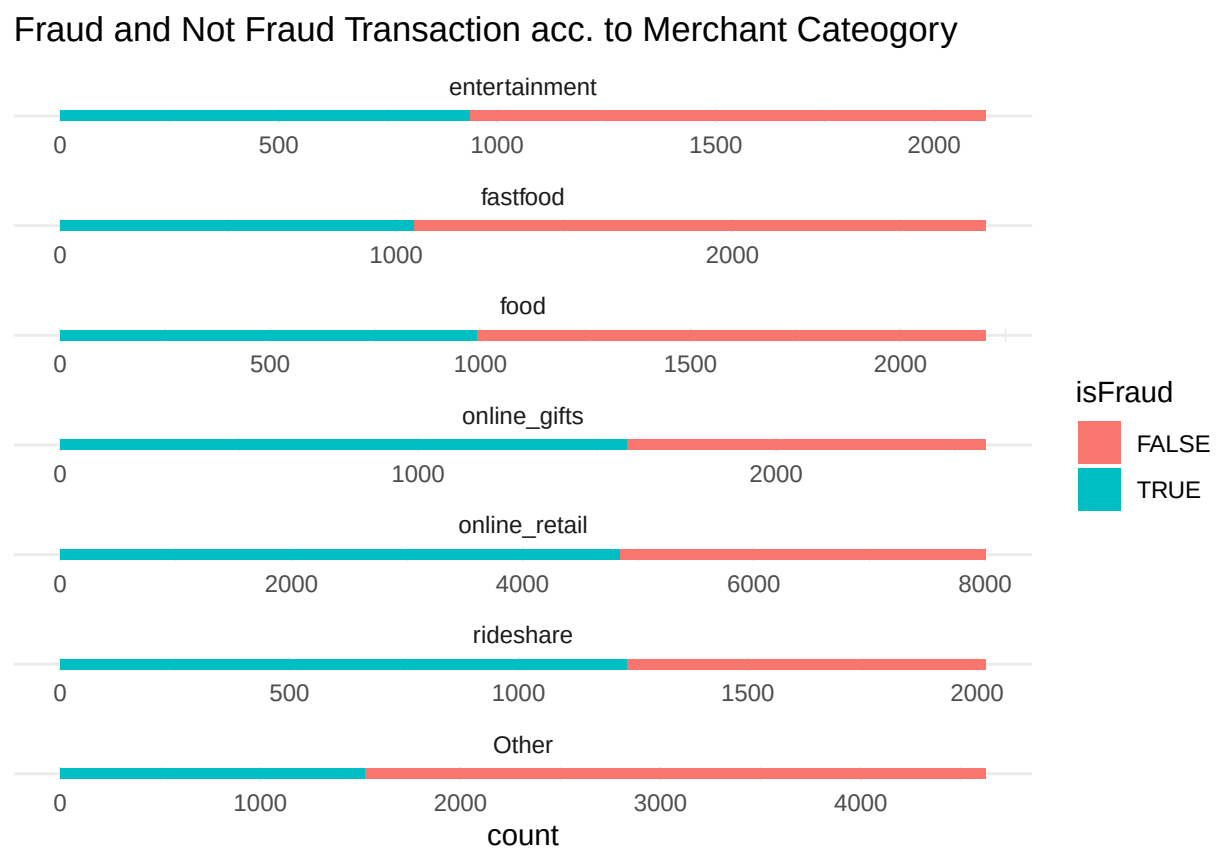
```
df_sample <-
df_sample %>%
  mutate(merchantCategoryCode = fct_lump(merchantCategoryCode,6))

df_sample %>%
  count(merchantCategoryCode,sort=T) %>%
  knitr::kable()
```

merchantCategoryCode	n
online_retail	8007
Other	4625
fastfood	2754
online_gifts	2586
food	2203

merchantCategoryCode	n
entertainment	2118
rideshare	2019

Online gifts, retail and rideshare are more effected by fraud

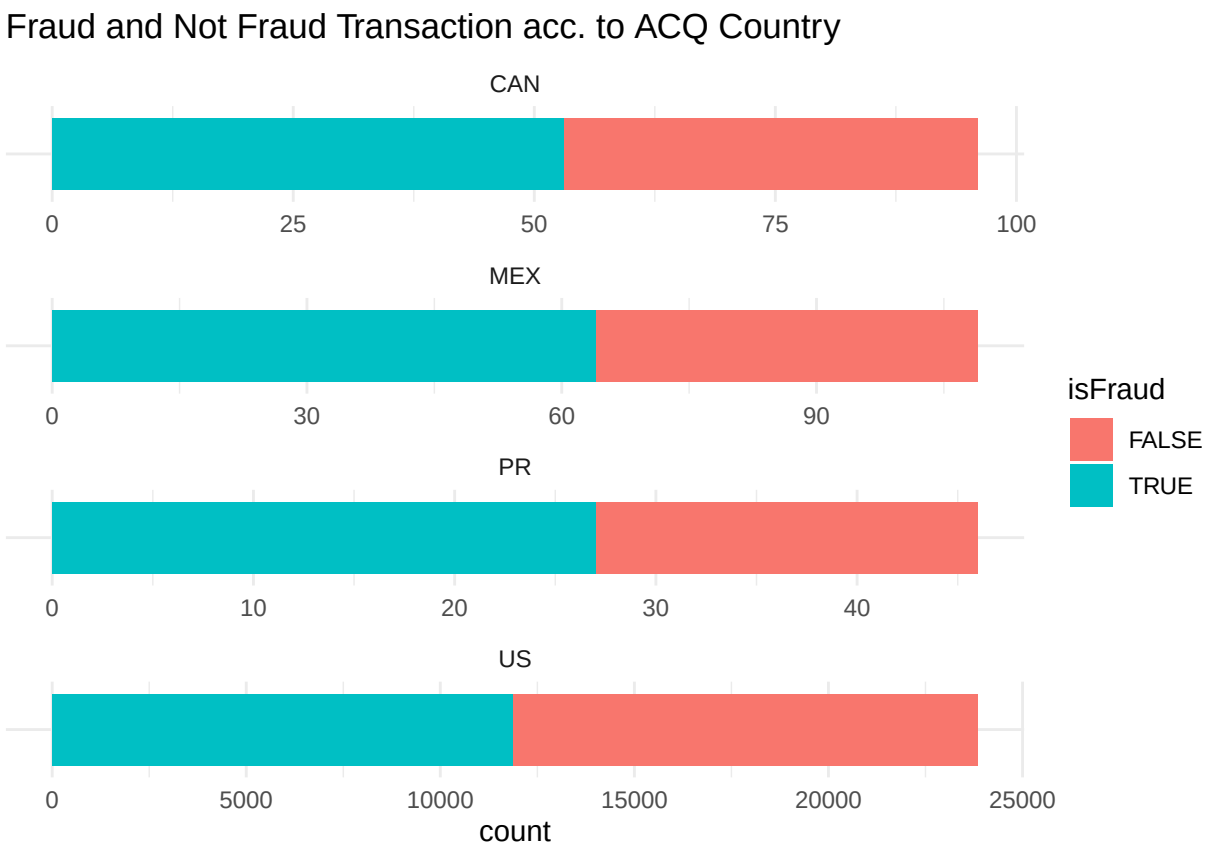
```
df_sample %>%
  ggplot(aes(merchantCategoryCode, fill=isFraud))+
  geom_bar()+
  coord_flip()+
  facet_wrap(~merchantCategoryCode, scales = "free", ncol = 1)+
  labs(
    title = "Fraud and Not Fraud Transaction acc. to Merchant Cateogory"
  )+
  theme_minimal()+
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



Countries ACQ Country

```
df_sample %>%
  filter(!is.na(acqCountry)) %>%
```

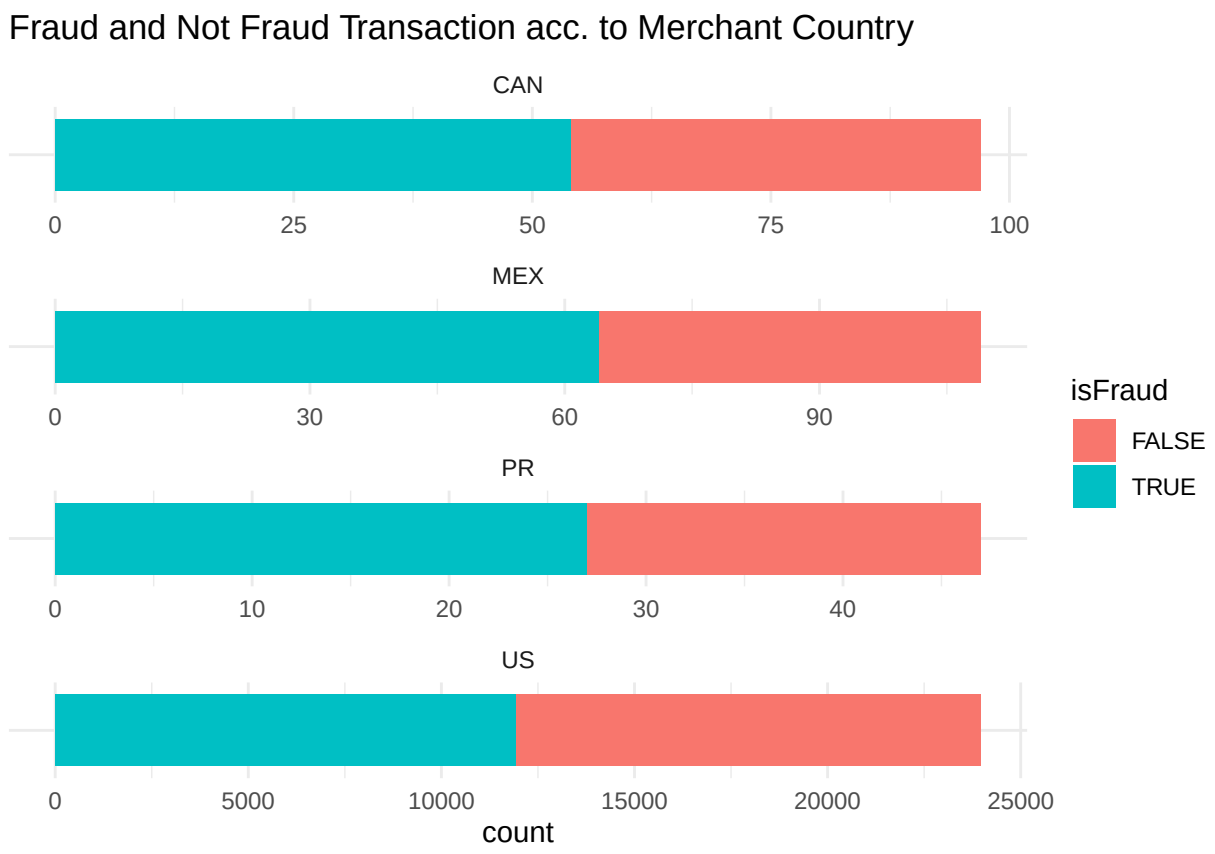
```
  ggplot(aes(acqCountry, fill=isFraud))+
  geom_bar()+
  coord_flip()+
  facet_wrap(~acqCountry, scales = "free", ncol = 1)+
  theme_minimal()+
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())+
  labs(
    title = "Fraud and Not Fraud Transaction acc. to ACQ Country"
  )+
  theme_minimal()+
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



Merchant Country Code

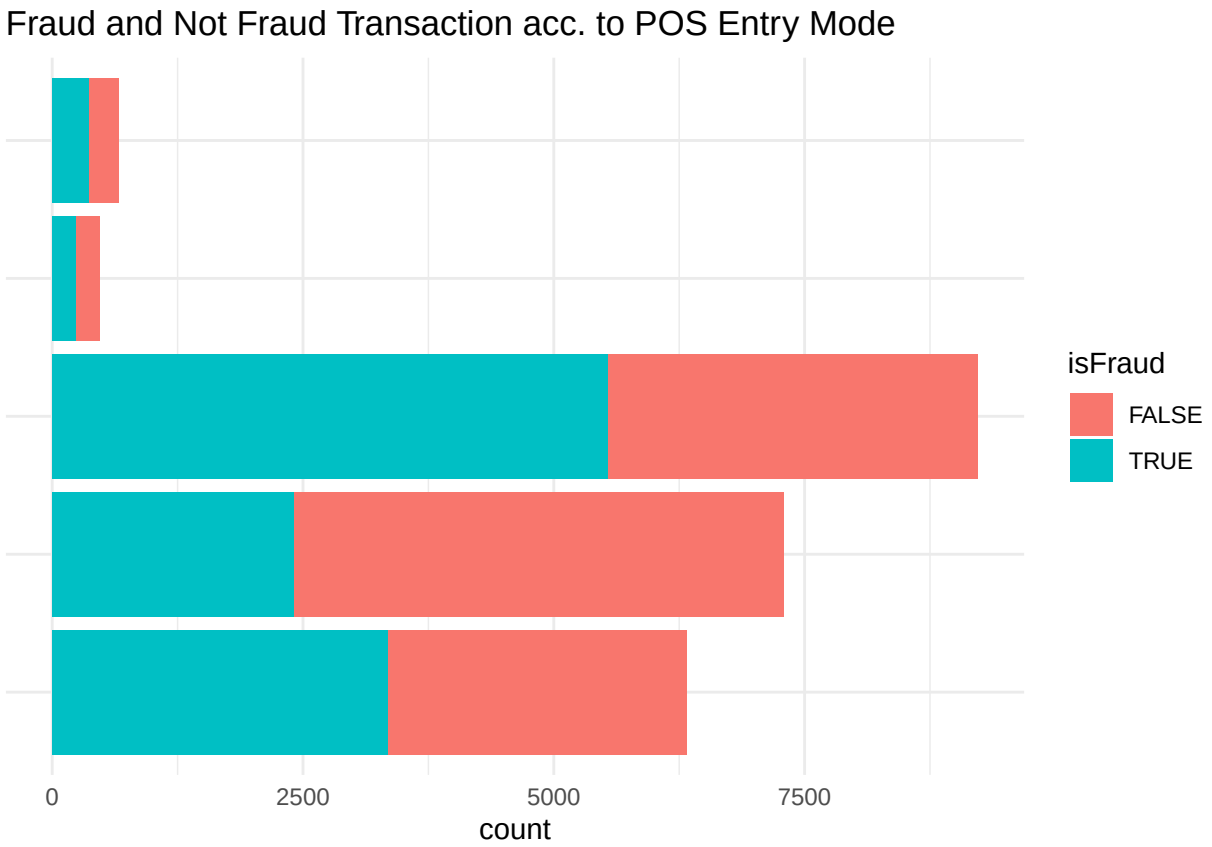
```
df_sample %>%
  filter(!is.na(merchantCountryCode)) %>%
  ggplot(aes(merchantCountryCode, fill=isFraud))+
  geom_bar()+
  coord_flip()+
  facet_wrap(~merchantCountryCode, scales = "free", ncol = 1)+
  labs(
    title = "Fraud and Not Fraud Transaction acc. to Merchant Country"
```

```
  )+
  theme_minimal()+
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



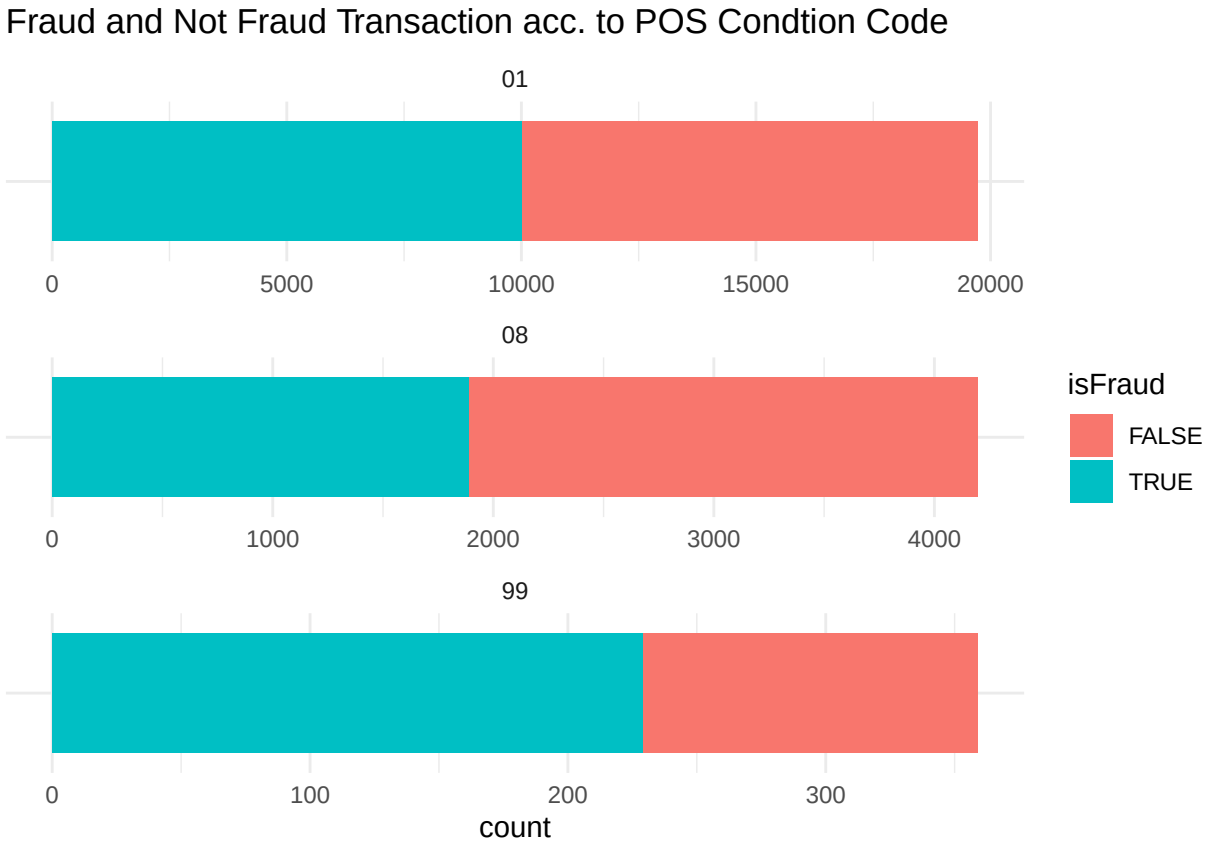
POS POS Entry Mode with 09 are more fraudulent and 05 more likely to be safe.

```
df_sample %>%
  filter(!is.na(posEntryMode)) %>%
  ggplot(aes(posEntryMode, fill=isFraud))+
  geom_bar()+
  coord_flip()+
  labs(
    title = "Fraud and Not Fraud Transaction acc. to POS Entry Mode"
  )+
  theme_minimal()+
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



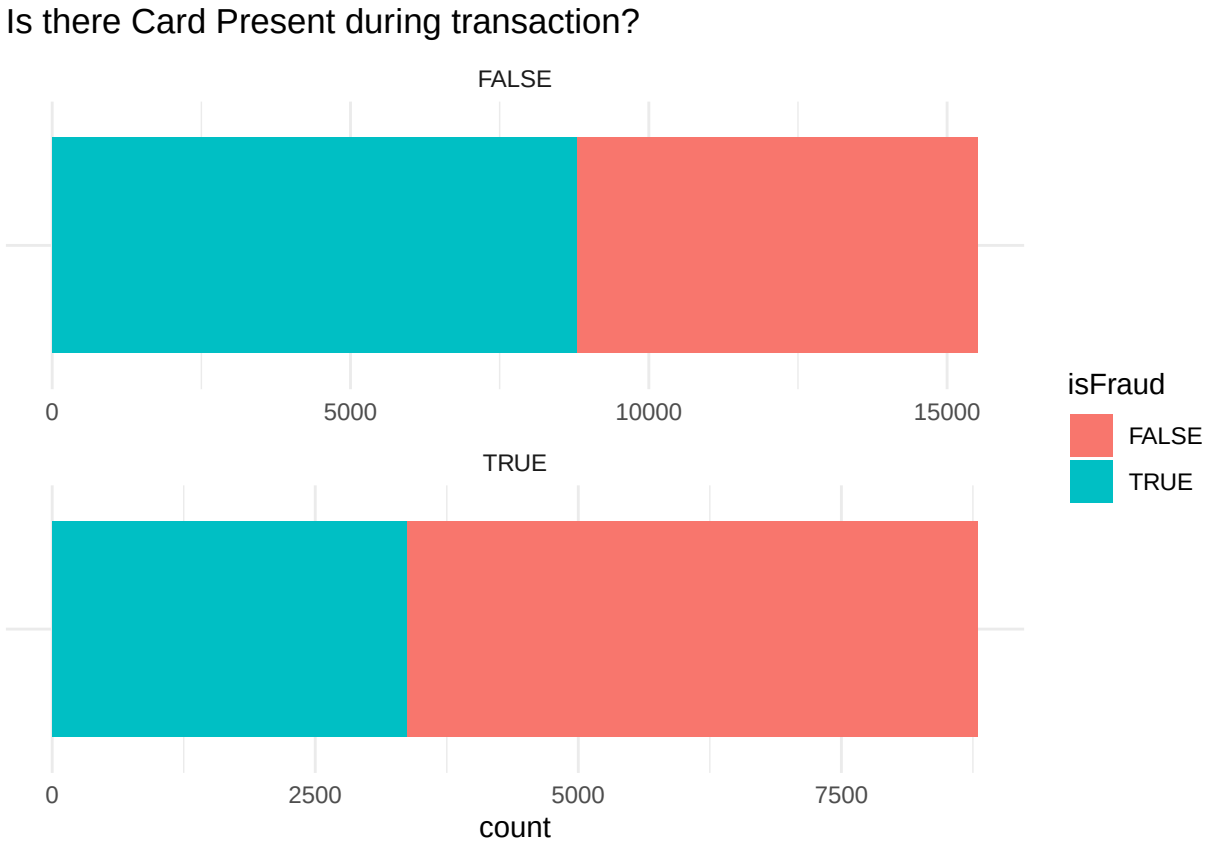
POS Condition Code with 99 are more likely to be fraudulent.

```
df_sample %>%
  filter(!is.na(posConditionCode)) %>%
  ggplot(aes(posConditionCode, fill=isFraud))+
  geom_bar()+
  coord_flip()+
  facet_wrap(~posConditionCode, scales = "free", ncol = 1)+
  labs(
    title = "Fraud and Not Fraud Transaction acc. to POS Condtion Code"
  )+
  theme_minimal()+
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



If card is physically present, it is less likely to be fraud.

```
df_sample %>%
  filter(!is.na(cardPresent)) %>%
  ggplot(aes(cardPresent, fill=isFraud))+
  geom_bar()+
  coord_flip()+
  facet_wrap(~cardPresent, scales = "free", ncol = 1)+
  labs(
    title = "Is there Card Present during transaction?"
  )+
  theme_minimal()+
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



Most of the values in expirationDateKeyInMatch is FALSE.

```
df_sample %>%
  count(expirationDateKeyInMatch)
```

```
## # A tibble: 2 x 2
##   expirationDateKeyInMatch     n
## * <lgl>                    <int>
## 1 FALSE                    24284
## 2 TRUE                        28
```

Taking everything in account, I have selected follow features:
1. acqCountry
2. creditLimit
3. availableMoney
4. transactionAmount
5. posEntryMode
6. posConditionCode
7. merchantCategoryCode
8. currentBalance
9. cardPresent

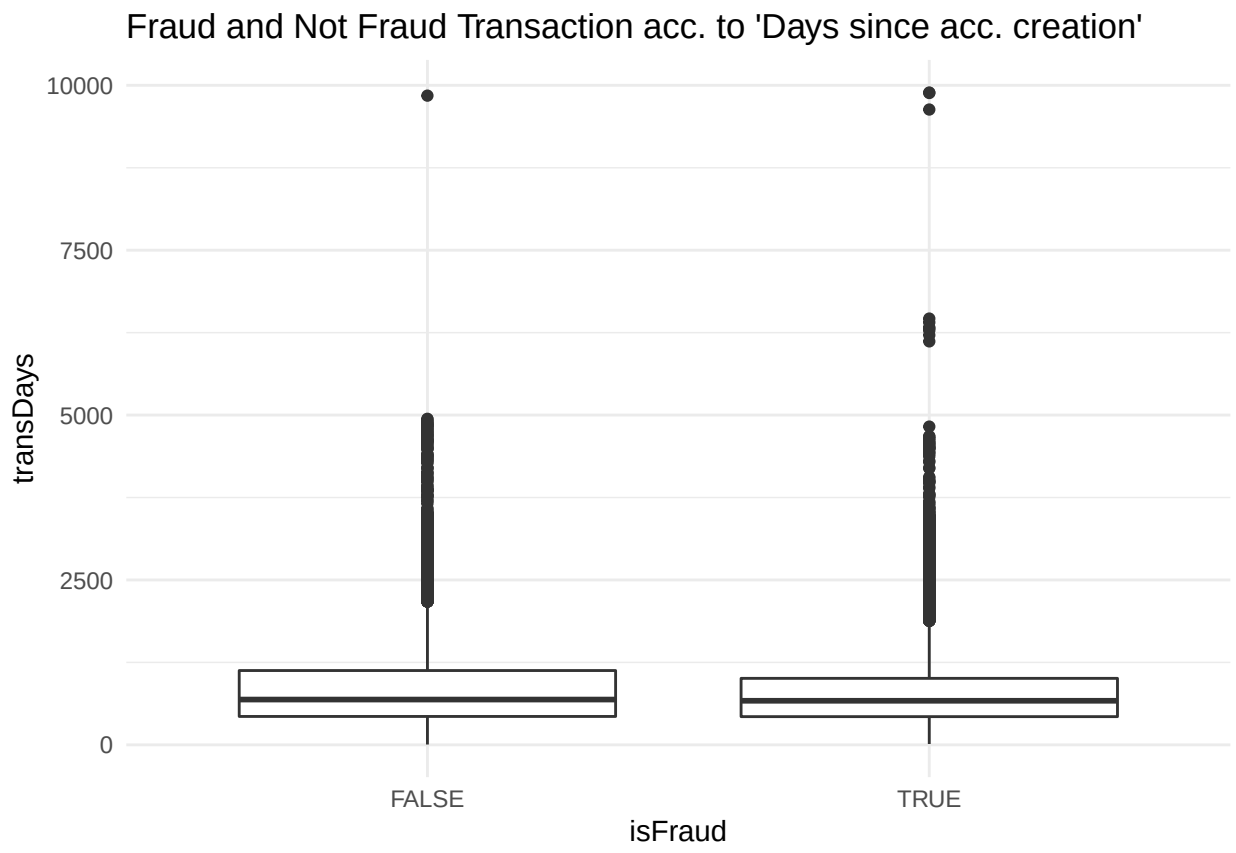
Feature Creation Lets create some features:
1. Transaction Days since acc. creation i.e. transDays (Are fraudulent transaction performed by card that are just created?)

2. Days remaining for expiry of the card i.e. expiryDays (Are older cards more likely to be safe?)
3. Does CVV match? i.e matchedCvv
4. Days since last address change. i.e. addressChangeDays (Cards whose address are recently changed, are used for fraudulent transaction ?)

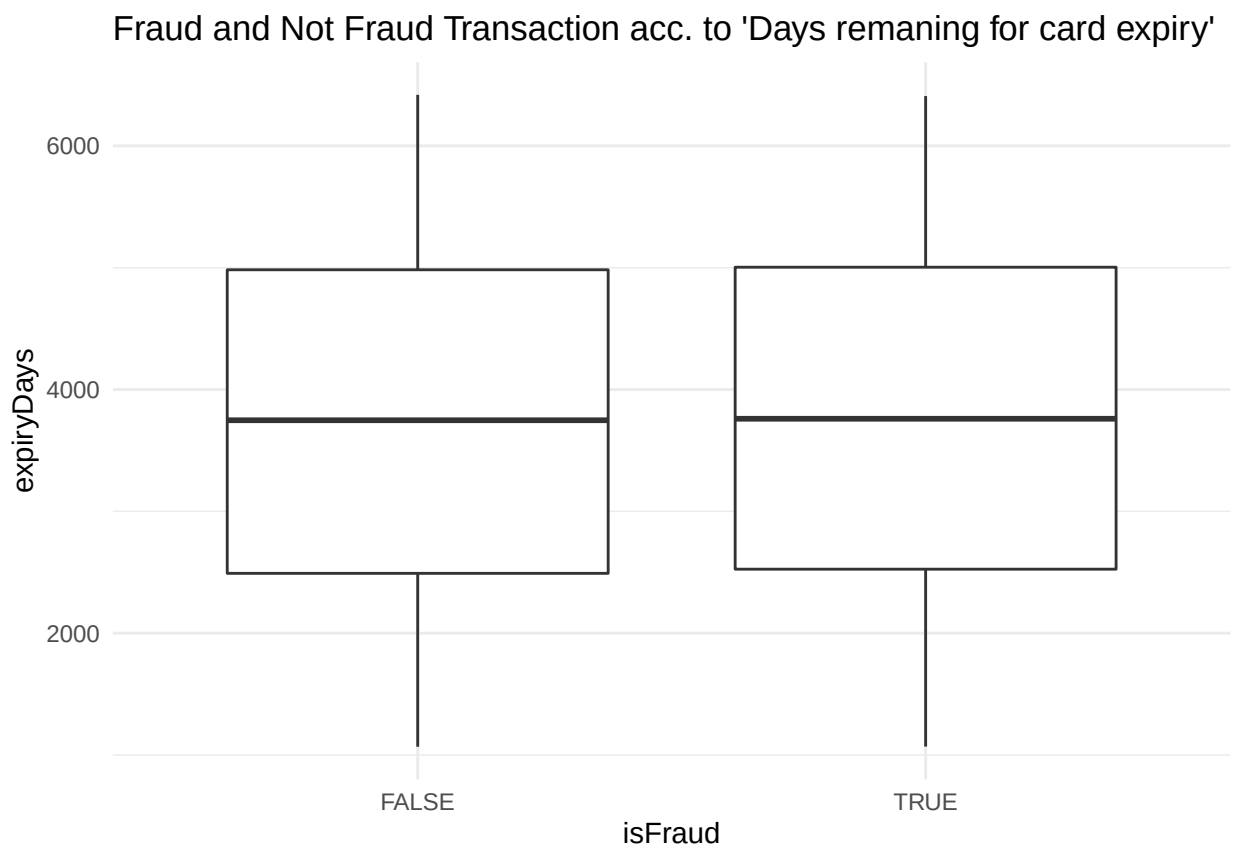
```
df_sample <-
df_sample %>%
  mutate(transDays = as.duration(accountOpenDate %--% transactionDateTime)/ddays(1),
         expiryDays = as.duration(transactionDateTime %--% currentExpDate)/ddays(1),
         addressChangeDays = as.duration(dateOfLastAddressChange %--% transactionDateTime)/ddays(1),
         matchedCvv = (cardCVV == enteredCVV)) %>%
  select(-dateOfLastAddressChange, -transactionDateTime, -accountOpenDate, currentExpDate,
         -cardCVV, enteredCVV)
```

Is there any significant difference in the distribution of Fraud and not Fraud from the feature created?

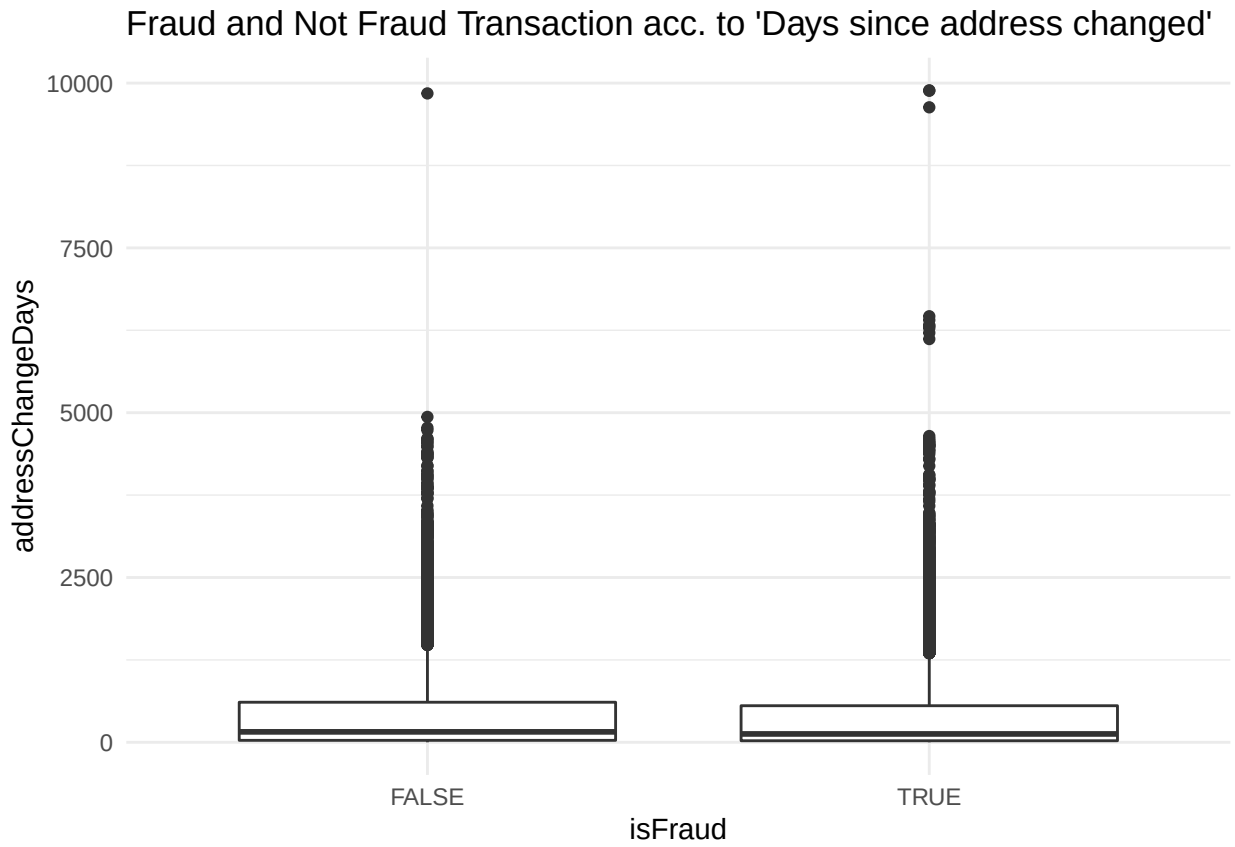
```
df_sample %>%
  ggplot(aes(transDays,isFraud))+
  geom_boxplot()+
  coord_flip()+
  labs(
    title = "Fraud and Not Fraud Transaction acc. to 'Days since acc. creation'"
  )+
  theme_minimal()
```



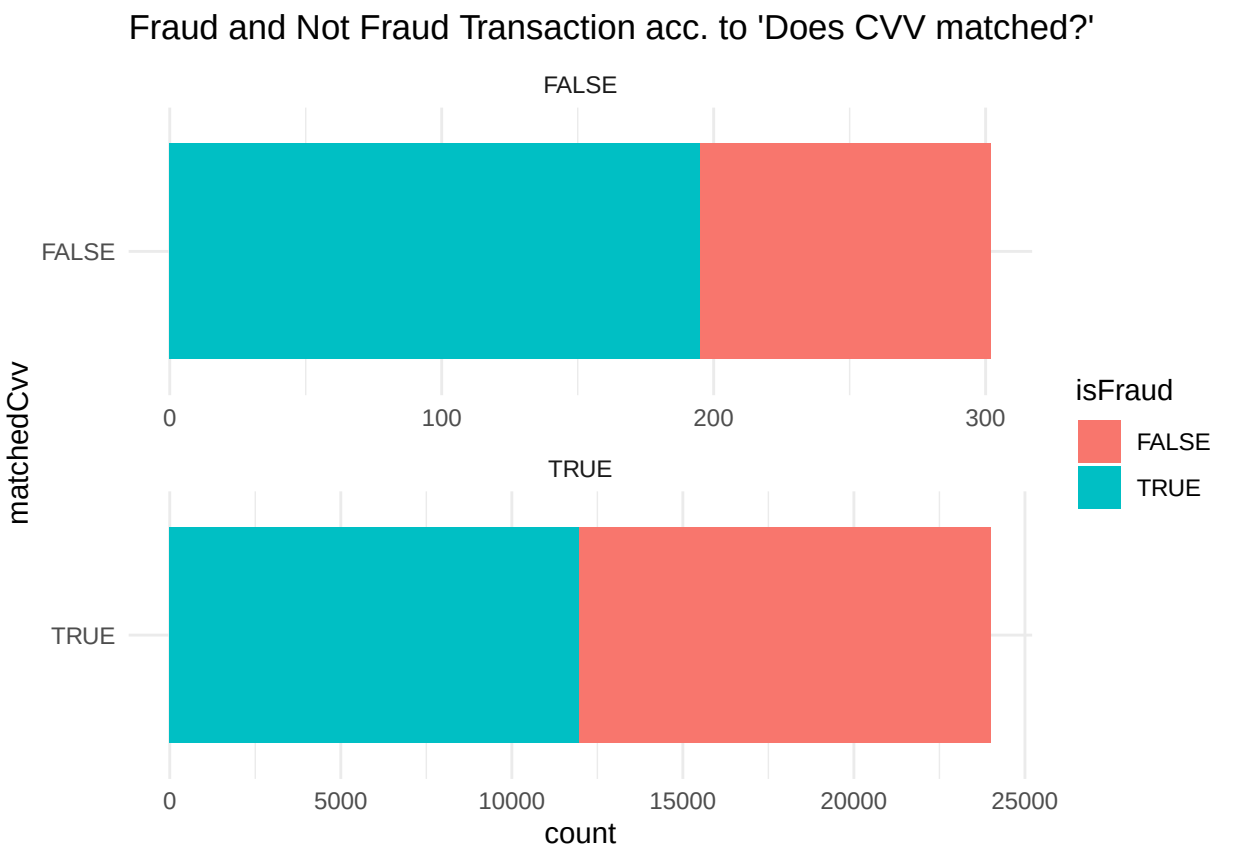
```
df_sample %>%
  ggplot(aes(expiryDays,isFraud))+
  geom_boxplot()+
  coord_flip()+
  labs(
    title = "Fraud and Not Fraud Transaction acc. to 'Days remaning for card expiry'"
  )+
  theme_minimal()
```




```
df_sample %>%
  ggplot(aes(addressChangeDays,isFraud))+
  geom_boxplot()+
  coord_flip()+
  labs(
    title = "Fraud and Not Fraud Transaction acc. to 'Days since address changed'"
  )+
  theme_minimal()
```




```
df_sample %>%
  ggplot(aes(matchedCvv, fill=isFraud))+
  geom_bar()+
  coord_flip()+
  facet_wrap(~matchedCvv, scales = "free", ncol = 1)+
  labs(
    title = "Fraud and Not Fraud Transaction acc. to 'Does CVV matched?'"
  )+
  theme_minimal()
```




```
# Too small data for non matched Cvv to make any judgment
```

Couldn't find any significant difference(visually) in the distribution of Fraud and not Fraud from the feature created. So dropping them.

```
df_sample <-
df_sample %>%
  select(-transDays, -expiryDays, -addressChangeDays, -matchedCvv)
```

Model Development

Dataset preperation

Here I am removing any row containing NA.

```
df_model <-
  df_sample %>%
  filter(transactionType == "PURCHASE") %>%
  na.omit() %>%
  select(creditLimit, availableMoney, transactionAmount, acqCountry,
         posEntryMode, posConditionCode, merchantCategoryCode, currentBalance,
         cardPresent, isFraud) %>%
  mutate_if(is.character, factor) %>%
  mutate(isFraud = factor(isFraud))
```

Our data to the model is

```
df_model %>%
  glimpse()
```

```
## Rows: 22,793
## Columns: 10
## $ creditLimit          <dbl> 20000, 5000, 20000, 5000, 20000, 5000, 10000, ...
## $ availableMoney       <dbl> 19975.20, 3395.86, 18364.67, 4184.65, 10089.18...
## $ transactionAmount    <dbl> 773.86, 61.18, 57.86, 212.19, 507.26, 233.91, ...
## $ acqCountry           <fct> US, US, US, US, US, US, US, US, US, US, US, US...
## $ posEntryMode         <fct> 09, 02, 05, 05, 02, 09, 90, 05, 05, 02, 02, 09...
## $ posConditionCode     <fct> 01, 08, 01, 08, 01, 01, 08, 01, 01, 01, 08, 01...
## $ merchantCategoryCode <fct> online_retail, online_retail, online_retail, e...
## $ currentBalance       <dbl> 24.80, 1604.14, 1635.33, 815.35, 9910.82, 3071...
## $ cardPresent          <lgl> FALSE, FALSE, FALSE, TRUE, TRUE, FALSE, TRUE, ...
## $ isFraud              <fct> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALS...
```

Splitting the data into test and train

```
library(tidymodels)
set.seed(2021)

df_split = initial_split(df_model)
df_train <- training(df_split)
df_test <- testing(df_split)
```

Also splitting training data into 10 fold cross validation for tuning our model.

```
df_train_fold = vfold_cv(df_train)

df_train_fold
```

```
## #  10-fold cross-validation
## # A tibble: 10 x 2
##    splits               id
##    <list>               <chr>
##  1 <split [15.4K/1.7K]> Fold01
##  2 <split [15.4K/1.7K]> Fold02
##  3 <split [15.4K/1.7K]> Fold03
##  4 <split [15.4K/1.7K]> Fold04
##  5 <split [15.4K/1.7K]> Fold05
##  6 <split [15.4K/1.7K]> Fold06
##  7 <split [15.4K/1.7K]> Fold07
##  8 <split [15.4K/1.7K]> Fold08
##  9 <split [15.4K/1.7K]> Fold09
## 10 <split [15.4K/1.7K]> Fold10
```

Preprocessor

Lets preprocesses the input data as:
1. Normalize all numeric data.
2. Create dummy variable for factors data.

```
preproc <-
recipe(isFraud ~ ., data = df_train) %>%
  step_normalize(all_numeric()) %>%
  step_dummy(all_nominal(), -isFraud)

preproc
```

```
## Data Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          9
##
## Operations:
##
## Centering and scaling for all_numeric()
## Dummy variables from all_nominal(), -isFraud
```

Model Definition (using Decision Tree)

I am selecting decision tree as model for fraud detection since decision tree are highly interpretable and we can easily see what attribute drives fraud detection the most.

```
tree_spec <- decision_tree(
  cost_complexity = tune(),
  tree_depth = tune(),
  min_n = tune()
) %>%
  set_engine("rpart") %>%
  set_mode("classification")

tree_spec
```

```
## Decision Tree Model Specification (classification)
##
## Main Arguments:
##   cost_complexity = tune()
##   tree_depth = tune()
##   min_n = tune()
##
## Computational engine: rpart
```

Workflow = preprocessor + model specification

```
tree_wf <-
  workflow() %>%
  add_recipe(preproc) %>%
  add_model(tree_spec)

tree_wf
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: decision_tree()
##
## -- Preprocessor ----------------------------------------------------------------
## 2 Recipe Steps
##
## * step_normalize()
## * step_dummy()
##
## -- Model -----------------------------------------------------------------------
## Decision Tree Model Specification (classification)
##
## Main Arguments:
##   cost_complexity = tune()
##   tree_depth = tune()
##   min_n = tune()
##
## Computational engine: rpart
```

Parameter Tuning

Tuning parameter : cost_complexity, tree_depth, min_n of our decision tree model in our 10 fold CV set. I am using regular grid search approach for parameter value.

Parameter values which we will be training the model on.

```
tree_grid <- grid_regular(cost_complexity(), tree_depth(), min_n(), levels = 3)

tree_grid
```

```
## # A tibble: 27 x 3
##    cost_complexity tree_depth min_n
##              <dbl>      <int> <int>
##  1    0.0000000001          1     2
##  2    0.00000316            1     2
##  3    0.1                   1     2
##  4    0.0000000001          8     2
##  5    0.00000316            8     2
##  6    0.1                   8     2
##  7    0.0000000001         15     2
##  8    0.00000316           15     2
##  9    0.1                  15     2
## 10    0.0000000001          1    21
## # ... with 17 more rows
```

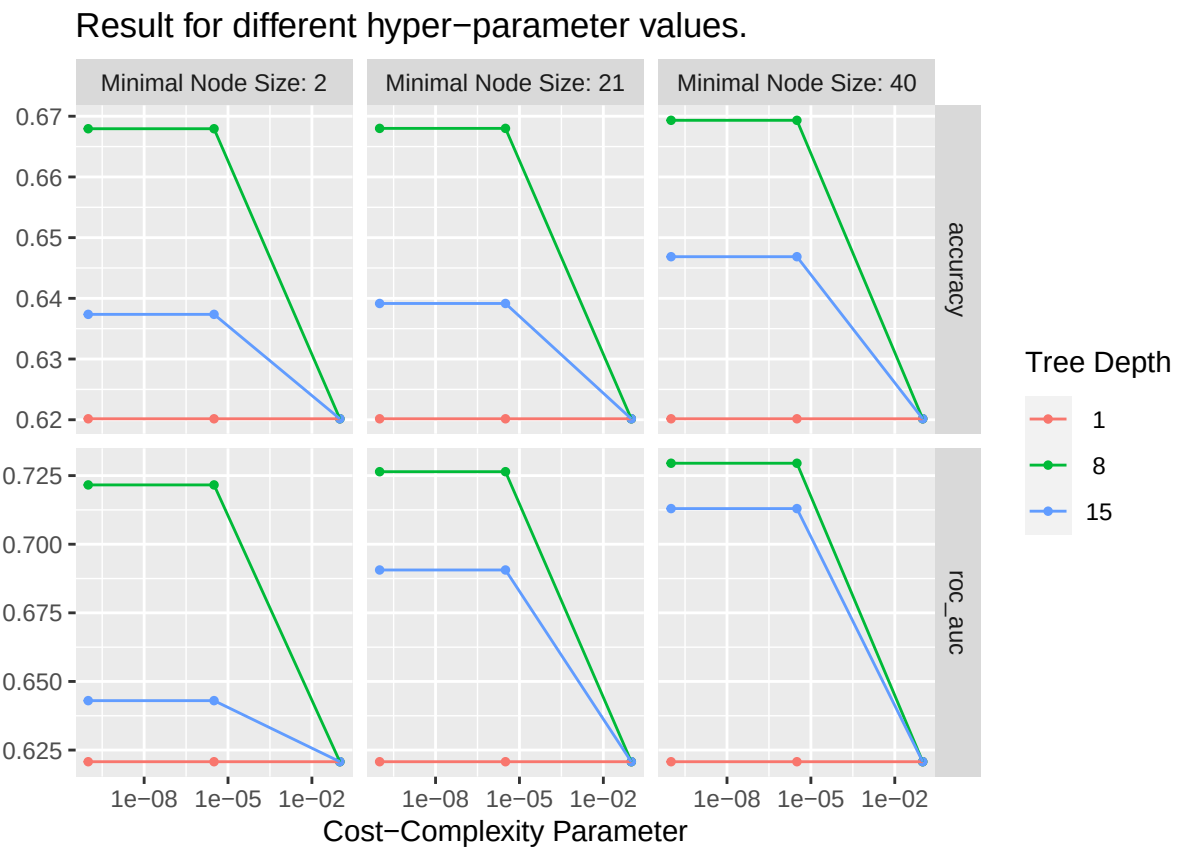
Training model on CV set.

```
doParallel::registerDoParallel()

tree_rs <-
  tree_wf %>%
  tune_grid(
```

```
  resamples = df_train_fold,
  grid = tree_grid
  )
```

Result of Tuning.

```
autoplot(tree_rs)+
  labs(title = "Result for different hyper-parameter values.")
```




```
tree_rs %>%
  collect_metrics() %>%
  select(-.config) %>%
  filter(.metric == "roc_auc") %>%
  arrange(desc(mean)) %>%
  head(10)
```

```
## # A tibble: 10 x 8
##    cost_complexity tree_depth min_n .metric .estimator  mean     n std_err
##              <dbl>      <int> <int> <chr>   <chr>      <dbl> <int>   <dbl>
##  1    0.0000000001          8    40 roc_auc binary     0.730    10 0.00326
##  2    0.00000316            8    40 roc_auc binary     0.730    10 0.00326
##  3    0.0000000001          8    21 roc_auc binary     0.726    10 0.00320
##  4    0.00000316            8    21 roc_auc binary     0.726    10 0.00320
##  5    0.0000000001          8     2 roc_auc binary     0.722    10 0.00295
```

```
##  6    0.00000316            8     2 roc_auc binary     0.722    10 0.00295
##  7    0.0000000001         15    40 roc_auc binary     0.713    10 0.00251
##  8    0.00000316           15    40 roc_auc binary     0.713    10 0.00251
##  9    0.0000000001         15    21 roc_auc binary     0.691    10 0.00390
## 10    0.00000316           15    21 roc_auc binary     0.691    10 0.00390
```

Fitting model.

```
final_tree <- finalize_workflow(tree_wf, select_best(tree_rs, "roc_auc"))

final_tree
```

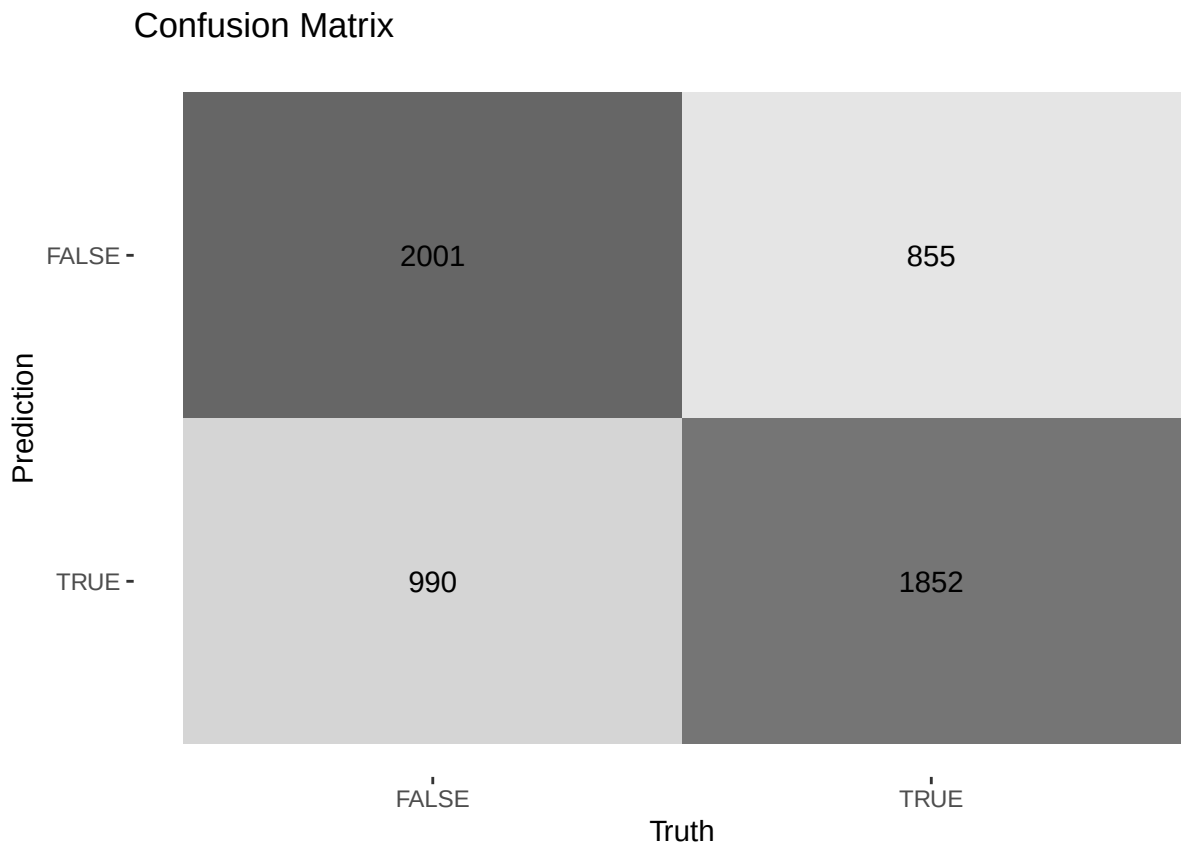
```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: decision_tree()
##
## -- Preprocessor ----------------------------------------------------------------
## 2 Recipe Steps
##
## * step_normalize()
## * step_dummy()
##
## -- Model -----------------------------------------------------------------------
## Decision Tree Model Specification (classification)
##
## Main Arguments:
##   cost_complexity = 1e-10
##   tree_depth = 8
##   min_n = 40
##
## Computational engine: rpart
```

```
final_result <- last_fit(final_tree, df_split)
```

Evaluating the Model

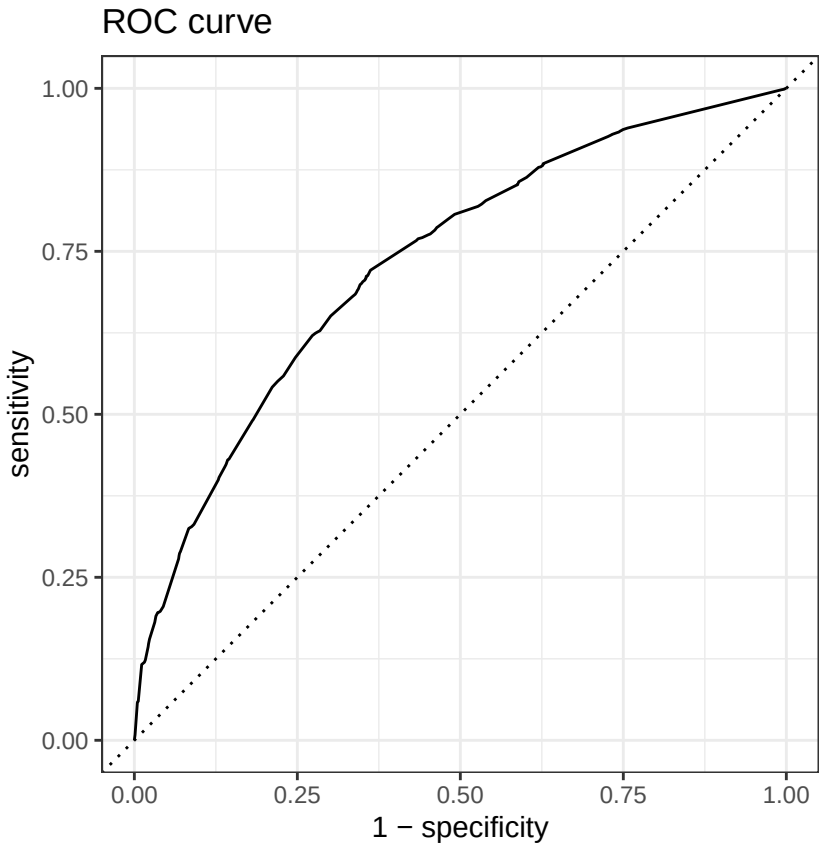
Confusion Matrix

```
library(yardstick)

final_result %>%
  collect_predictions() %>%
  yardstick::conf_mat(.pred_class, isFraud) %>%
  autoplot(type = "heatmap") +
  labs(
    title = "Confusion Matrix"
  )
```



ROC curve

```
final_result %>%
  collect_predictions() %>%
  roc_curve(isFraud, .pred_FALSE)%>%
  autoplot() +
  labs(
    title = "ROC curve"
  )
```



Metrics

```
final_result %>%
  collect_predictions() %>%
  metrics(.pred_class, isFraud)
```

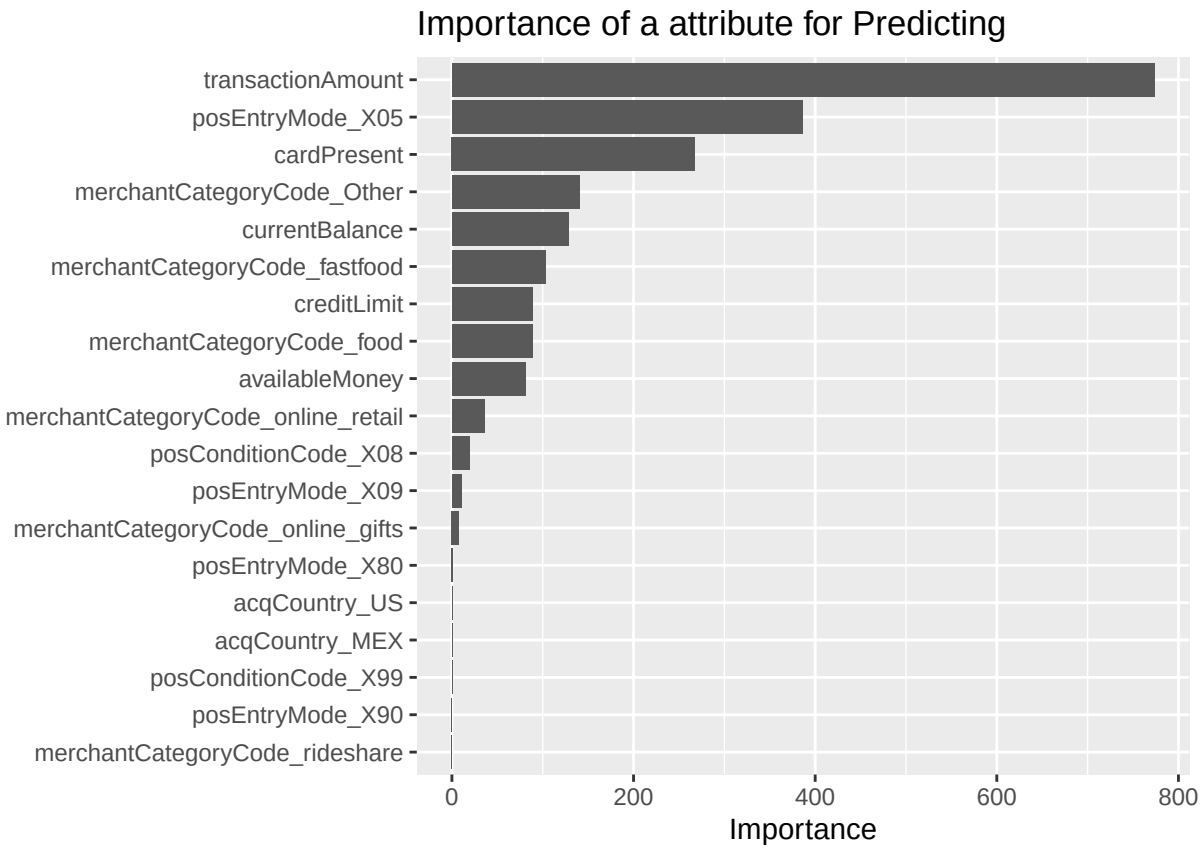
```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.676
## 2 kap      binary         0.352
```

Variable Importance

Transaction Amount and Merchant category are important predictor for detecting fraudulent transactions.

```
library(vip)

fit(final_tree, df_train)%>%
  pull_workflow_fit() %>%
  vi() %>%
  ggplot(aes(Importance, fct_reorder(Variable,Importance))) +
```

```
geom_col() +
labs(
  title = "Importance of a attribute for Predicting",
  y= ""
)
```



Remarks

Methods I attempted that didn't work
* I tried creating features(transDays, expiryDays, addressChangeDays, matchedCvv) but I couldn't find any significant difference(visually) in the distribution of Fraud and not Fraud from the feature created. So I dropped it.

I would have tried following ideas if I had more time
1. What if purchase and reversal don't follow immediately? For eg, person X bought A and B. Than after a minute X canceled A.
2. Can 10 minute be considered 'short duration' in multi-swipe transaction. I would try and test different duration.
3. Instead of dropping row containing missing values, I will look into imputing it.
4. Balancing the dataset without under sampling. For eg, using Oversampling method such as SMOTE or look into Cost-Sensitive Learning.
5. Interaction between attributes. For eg, interaction between country and POS entry mode, which would take into account: Does POS entry mode of 09 from Mexico more likely to be fraudulent?
6. I would look into more complex algorithm for modeling such as SVM or Ensemble learning.

Some Question I had.
* Is it possible for current balance to be more than available? Is it due to overdraft?

```
df %>%
  mutate( availableGTcurrent = availableMoney>currentBalance) %>%
  count(availableGTcurrent) %>%
  ggplot(aes(availableGTcurrent,n))+
  geom_col()+
  labs(
    title = "Available Money > Current Balance",
    x = "",
    y = "count"
  )+
  theme_minimal()
```