Challenge 2 R Notebook

Eagle Xuhui Ying

10/31/2022

Libraries

```
options(warn = -1)
library(tidyverse)   # tidyverse
library(tidymodels)  # modeling interface
library(janitor)     # clean_names()
library(skimr)       # profiling
library(vip)         # variable importance
```

1. Import Data

```
fraud <- read_csv("project_2_training.csv") %>% clean_names()
head(fraud)
```

event_id <dbl>	account_age_days <dbl>	transaction_amt <dbl>	transaction_adj_amt <dbl>	historic_velocity <dbl>	ip_address <chr>
2608000	5774	2172	78	4523	81.237.240.91
214500	5405	2887	51	4057	160.32.216.178
294600	6570	2887	56	5602	109.156.235.28

event_id <dbl>	account_age_days <dbl>	transaction_amt <dbl>	transaction_adj_amt <dbl>	historic_velocity <dbl>	ip_address <chr>
477060	3865	1591	58	3926	158.152.249.45
1993100	6625	3297	70	6675	167.164.91.31
1810560	5513	2946	56	6196	199.82.244.185

6 rows | 1-6 of 27 columns

```
fraud_kaggle <- read_csv("project_2_holdout.csv") %>% clean_names()
head(fraud_kaggle)
```

event_id <dbl>	account_age_days <dbl>	transaction_amt <dbl>	transaction_adj_amt <dbl>	historic_velocity <dbl>	ip_address <chr>
109420	4462	3648	54	6325	197.108.209.59
1049060	3360	2180	54	4030	178.166.190.20
2805320	4725	2063	60	4372	163.48.86.34
450840	5336	2319	72	4385	55.233.5.10
423160	6115	3413	61	5942	119.228.12.223
278980	3664	2241	52	3897	161.16.159.136

6 rows | 1-6 of 27 columns

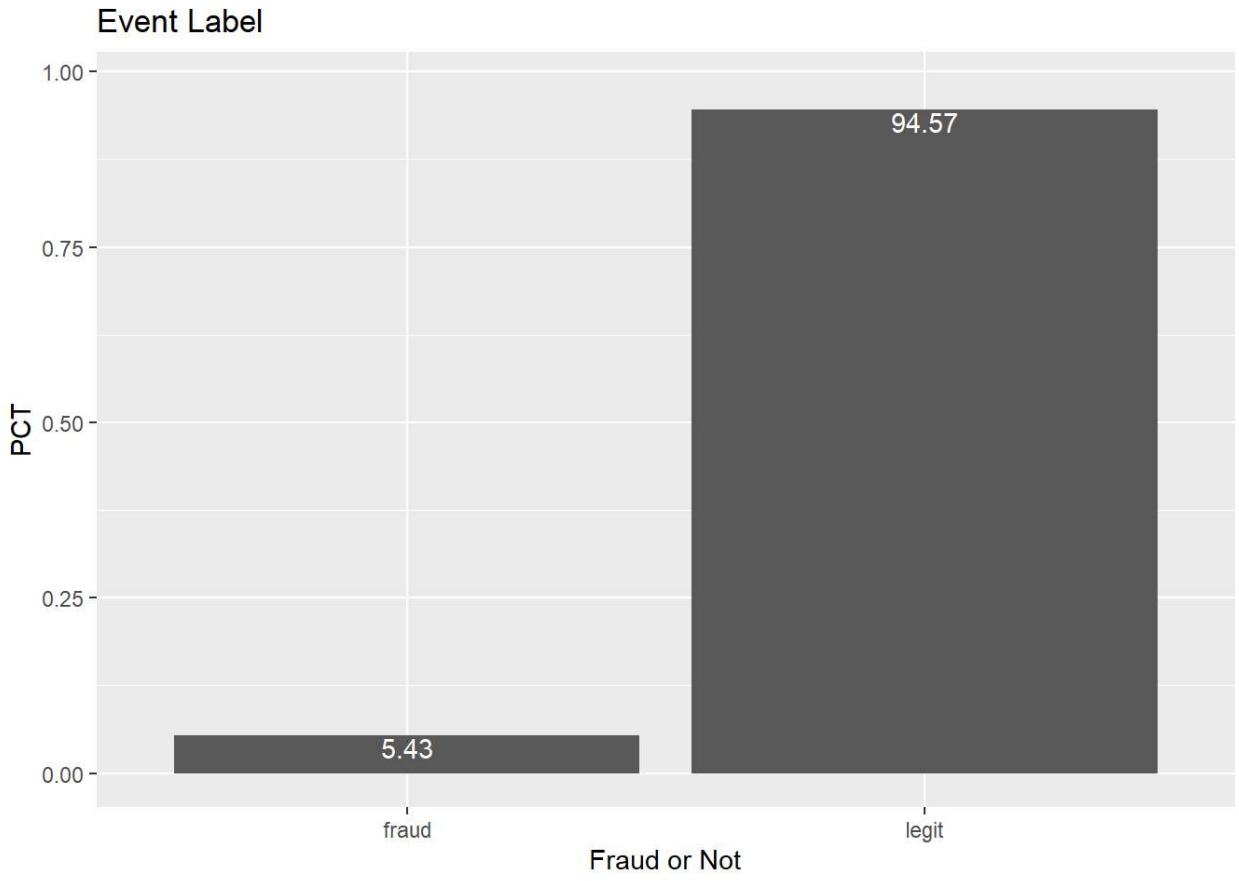
2. Explore Target

```
fraud_summary <- fraud %>%
  count(event_label) %>%
  mutate(pct = n/sum(n))

fraud_summary
```

event_label <chr>	n <int>	pct <dbl>
fraud	6785	0.05428
legit	118215	0.94572

2 rows

```
fraud_summary %>%
  ggplot(aes(x=factor(event_label),y=pct)) +
  geom_col()  +
  geom_text(aes(x=factor(event_label),y=pct+0.034, label = round(pct*100,2)), vjust = 2.25, colour = "white") +
  labs(title="Event Label", x="Fraud or Not", y="PCT")
```



3. Explore your data

```
fraud %>% skimr::skim_without_charts()
```

Data summary

Name	Piped data
Number of rows	125000
Number of columns	27
_____	
Column type frequency:	
character	18
numeric	9
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
ip_address	0	1	3	15	0	13314	0
user_agent	0	1	3	147	0	8571	0
email_domain	0	1	3	25	0	6992	0
phone_number	0	1	3	22	0	11906	0
billing_city	0	1	3	24	0	8980	0
billing_state	0	1	3	14	0	51	0
currency	0	1	3	3	0	4	0
cvv	0	1	1	3	0	26	0
signature_image	0	1	1	3	0	27	0
transaction_type	0	1	1	3	0	27	0
transaction_env	0	1	1	3	0	27	0
event_timestamp	90	1	13	16	0	111339	0
applicant_name	124	1	6	27	0	84958	0
billing_address	111	1	11	38	0	124884	0
merchant_id	89	1	11	11	0	124904	0
locale	115	1	5	6	0	293	0
tranaction_initiate	100	1	1	1	0	26	0
event_label	0	1	5	5	0	2	0

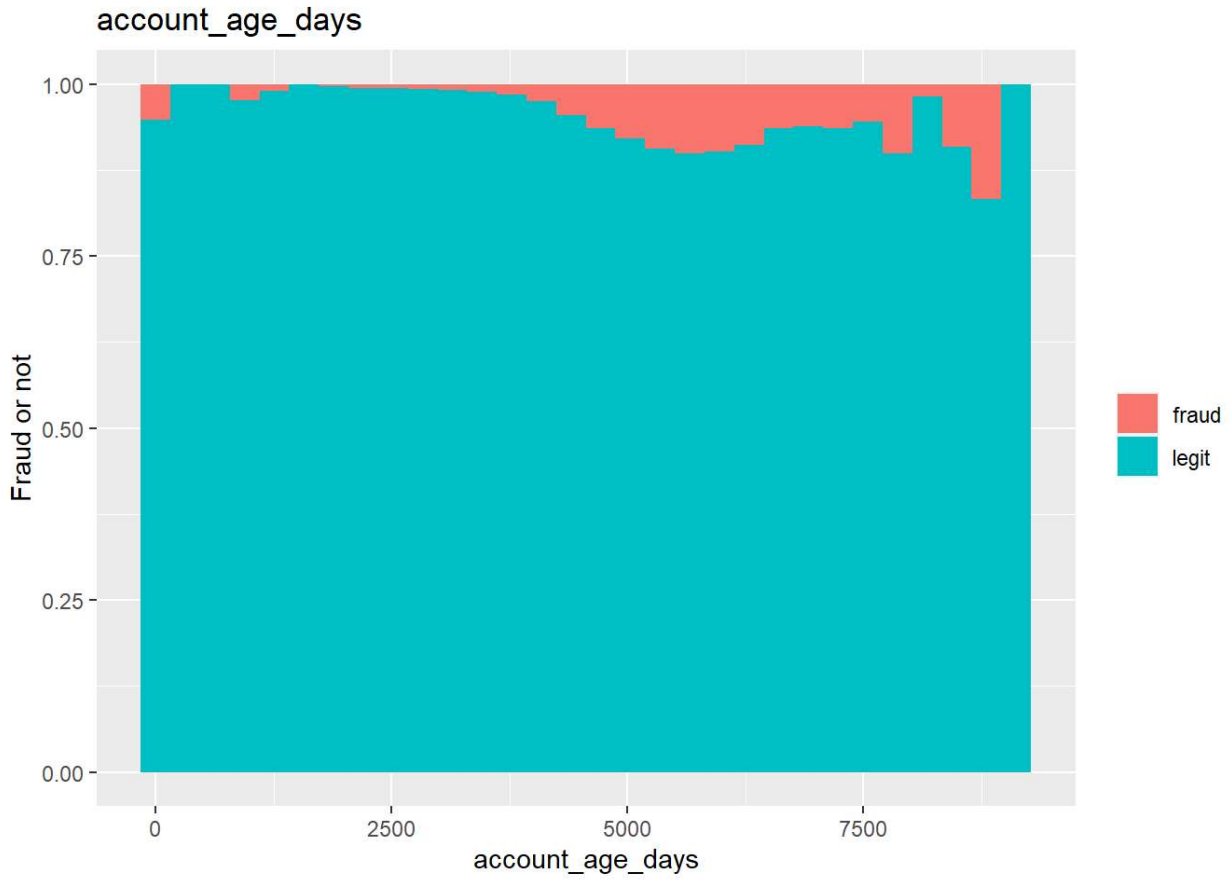
Variable type: numeric

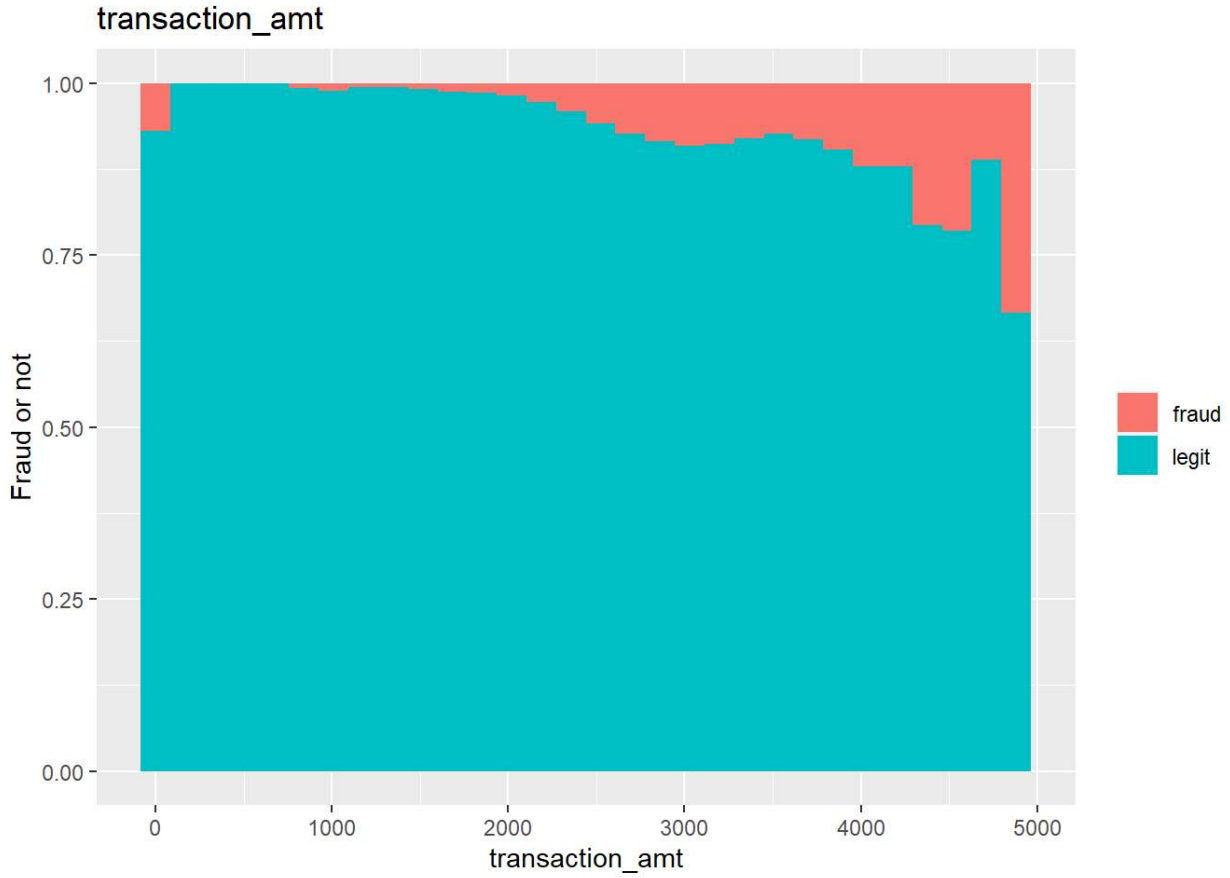
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
event_id	0	1	1500443.74	866356.54	20	750535	1500570	2251825.00	2999960
account_age_days	0	1	4642.45	1160.92	-1	3822	4668	5472.00	9119
transaction_amt	0	1	2519.55	609.30	-1	2102	2543	2952.00	4880
transaction_adj_amt	0	1	54.14	10.17	-1	48	55	61.00	99
historic_velocity	0	1	4699.90	1194.36	-1	3871	4731	5549.00	8875
billing_postal	98	1	50210.79	28405.88	503	25298	50124	74457.00	99950
card_bin	110	1	41813.29	10084.07	6040	35378	42061	47330.75	67639
days_since_last_logon	113	1	49.81	29.22	0	24	50	75.00	100
inital_amount	109	1	7999.64	4050.18	1000	4486	8007	11498.00	15000

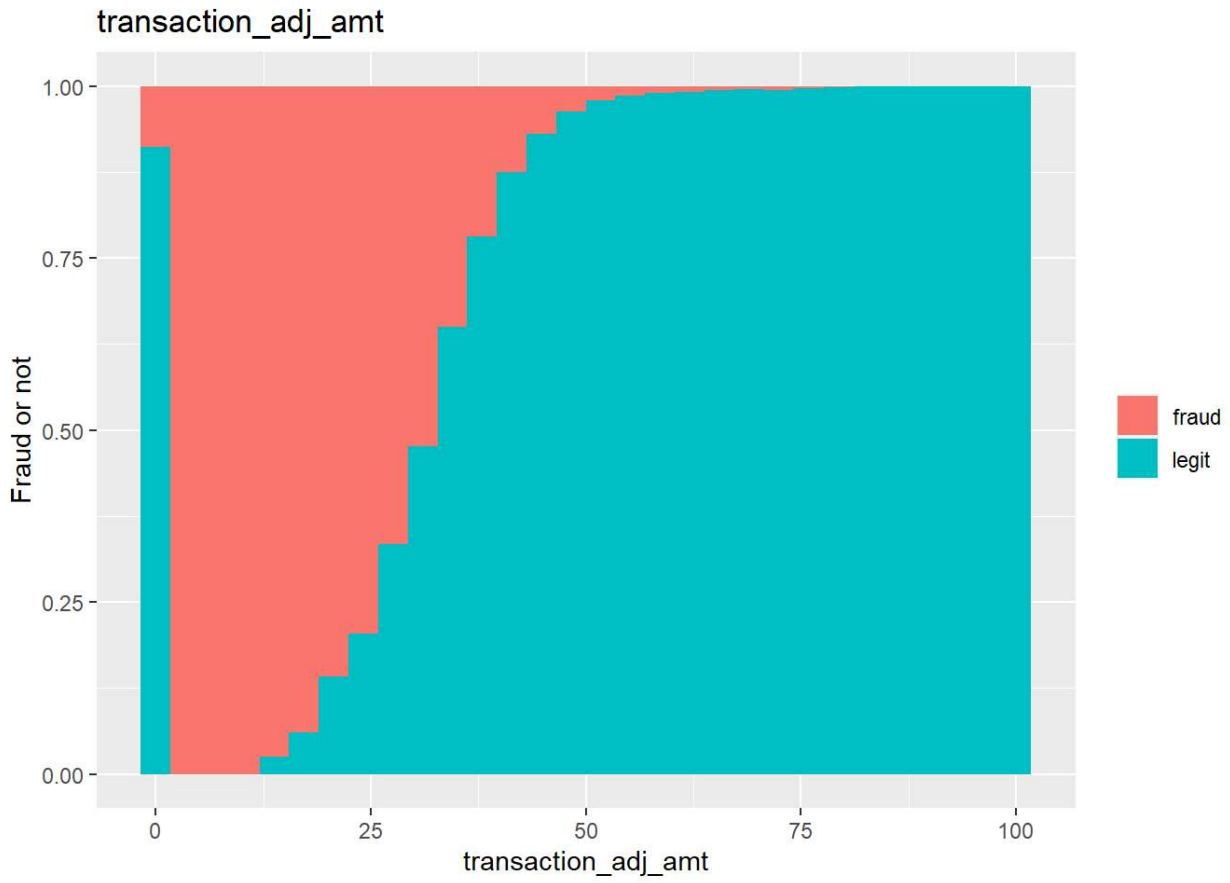
Explore numerics

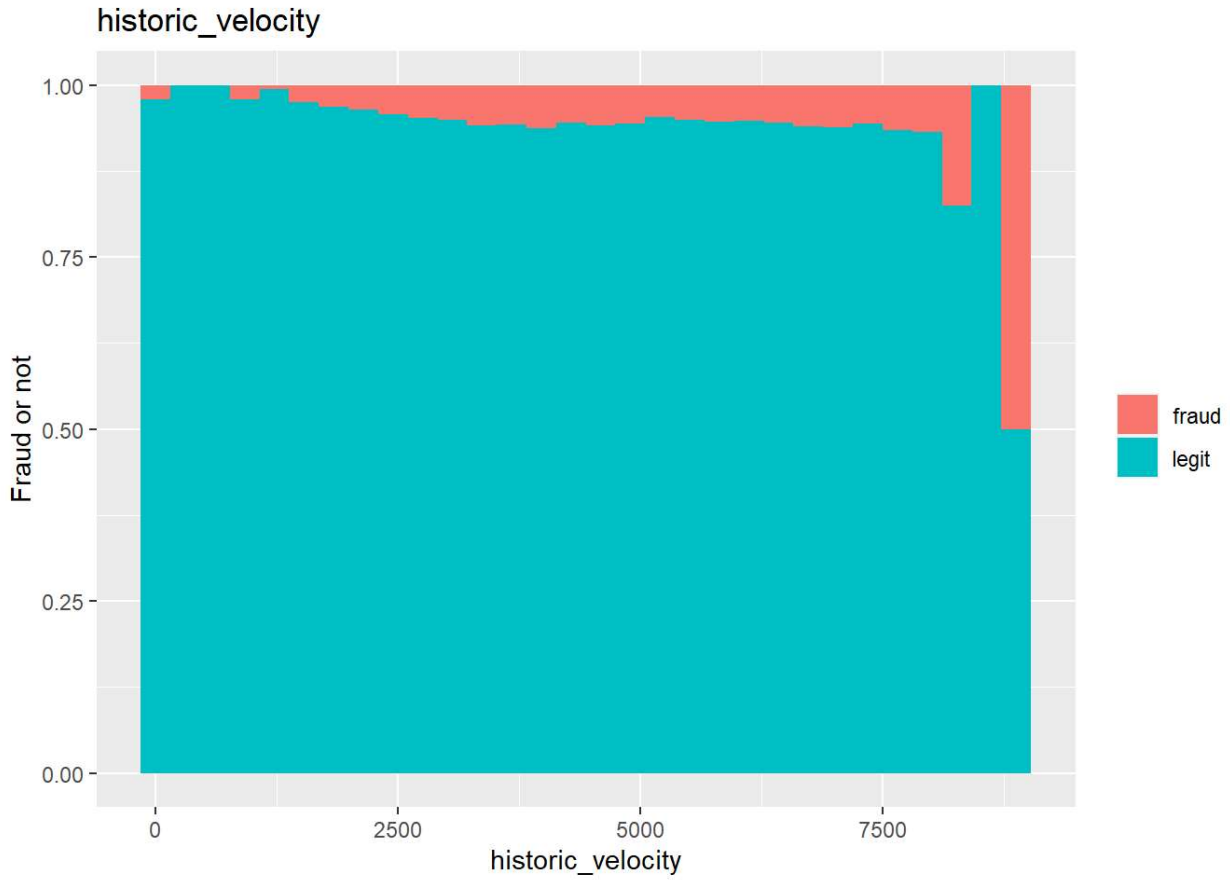
numeric variables: account_age_days, transaction_amt, transaction_adj_amt, historic_velocity, days_since_last_logon, inital_amount

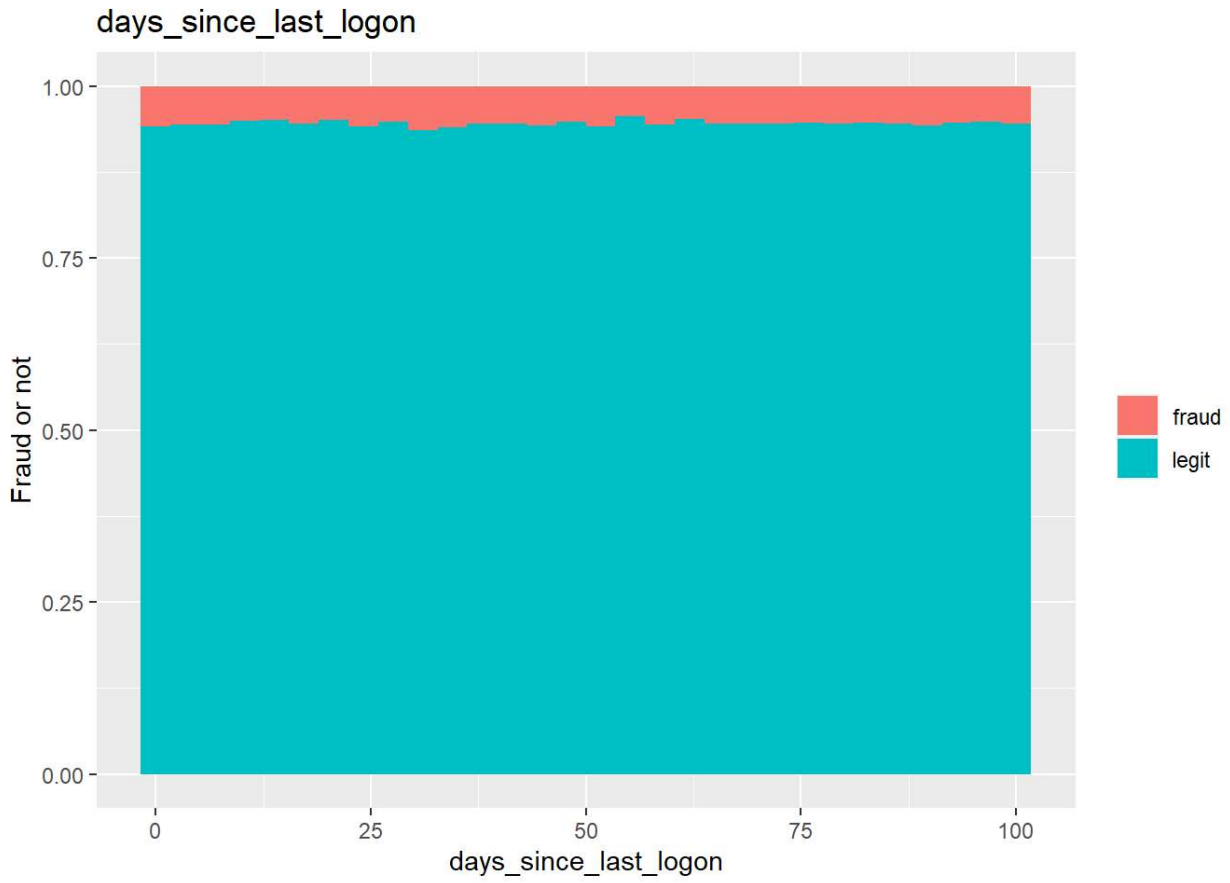
```
histogram_fill <- function(m){
    fraud %>%
    na.omit() %>%
    ggplot(aes(x=!!as.name(m), fill=as.factor(event_label))) +
    geom_histogram(position = 'fill') +
    labs(title = as.character(m), y = 'Fraud or not') +
    theme(legend.title = element_blank())
}

histogram_stack <- function(m){
    fraud %>%
    na.omit() %>%
    ggplot(aes(x=!!as.name(m), fill=as.factor(event_label))) +
    geom_histogram(position = 'stack') +
    labs(title = as.character(m), y = 'Fraud or not') +
    theme(legend.title = element_blank())
}

numerics <- c('account_age_days', 'transaction_amt', 'transaction_adj_amt', 'historic_velocity', 'days_since_last_log
          on', 'inital_amount')

for (c in numerics){
    print(histogram_fill(c))
}
```

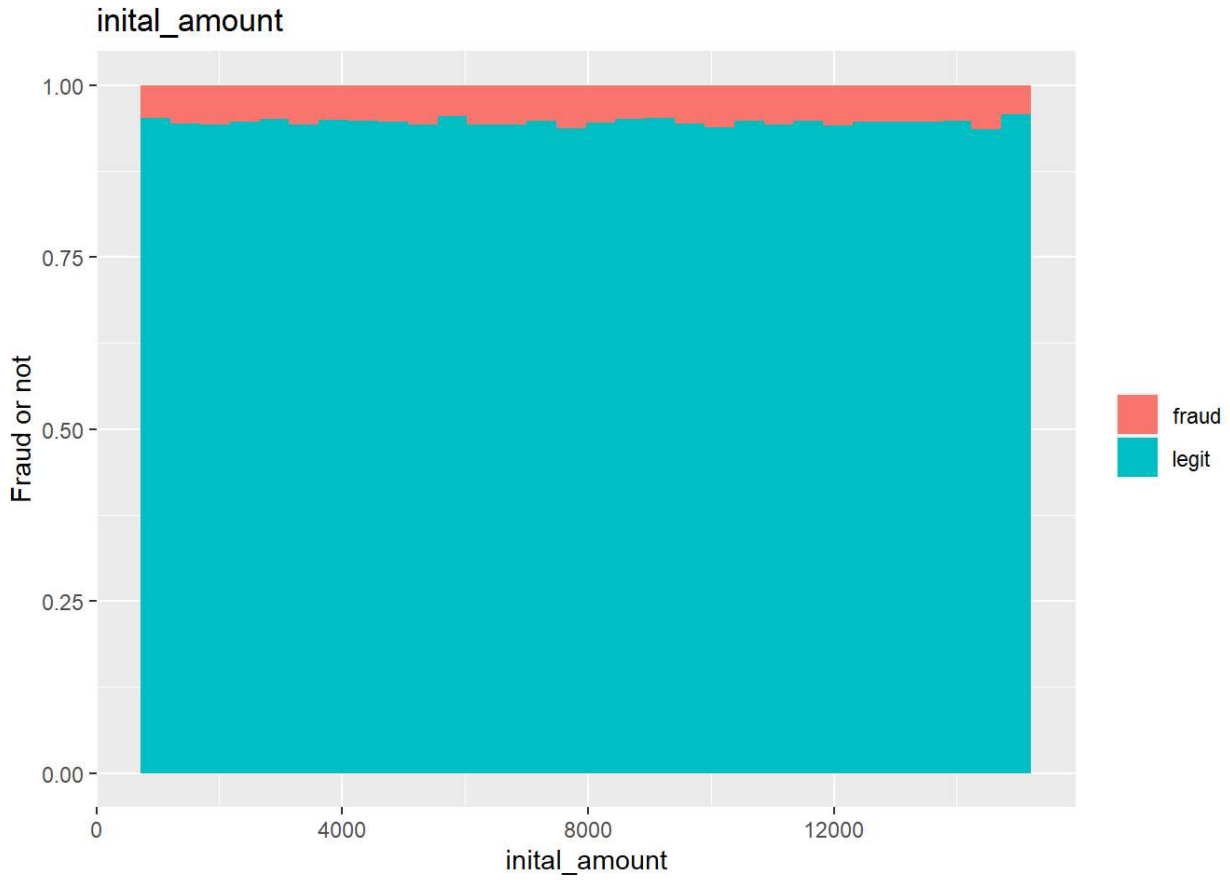

account_age_days
1.00
0.75
0.50
0.25
0.00
Fraud or not
0
2500
5000
7500
account_age_days
fraud
legit


transaction_amt
1.00
0.75
0.50
0.25
0.00
Fraud or not
0
1000
2000
3000
4000
5000
transaction_amt
fraud
legit

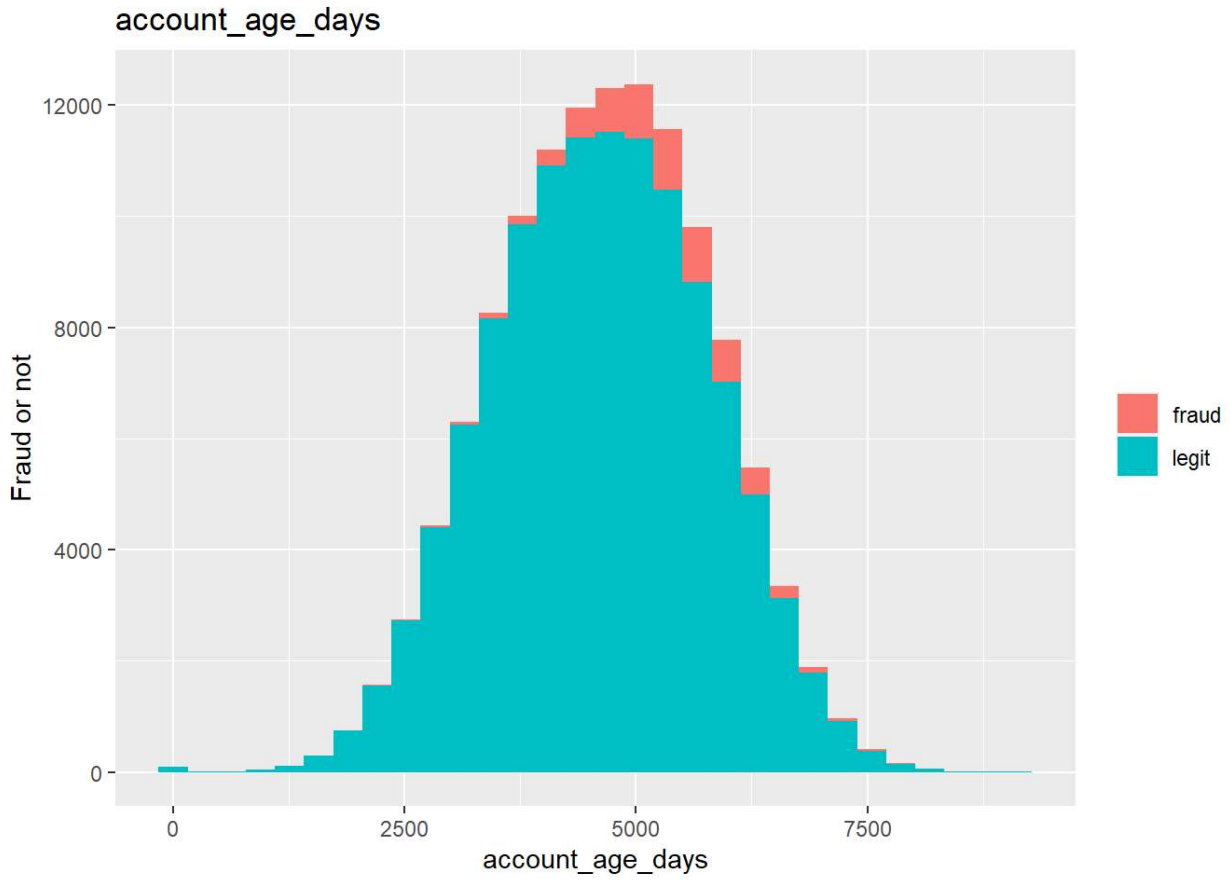

transaction_adj_amt
1.00
0.75
0.50
0.25
0.00
Fraud or not
0
25
50
75
100
transaction_adj_amt
fraud
legit

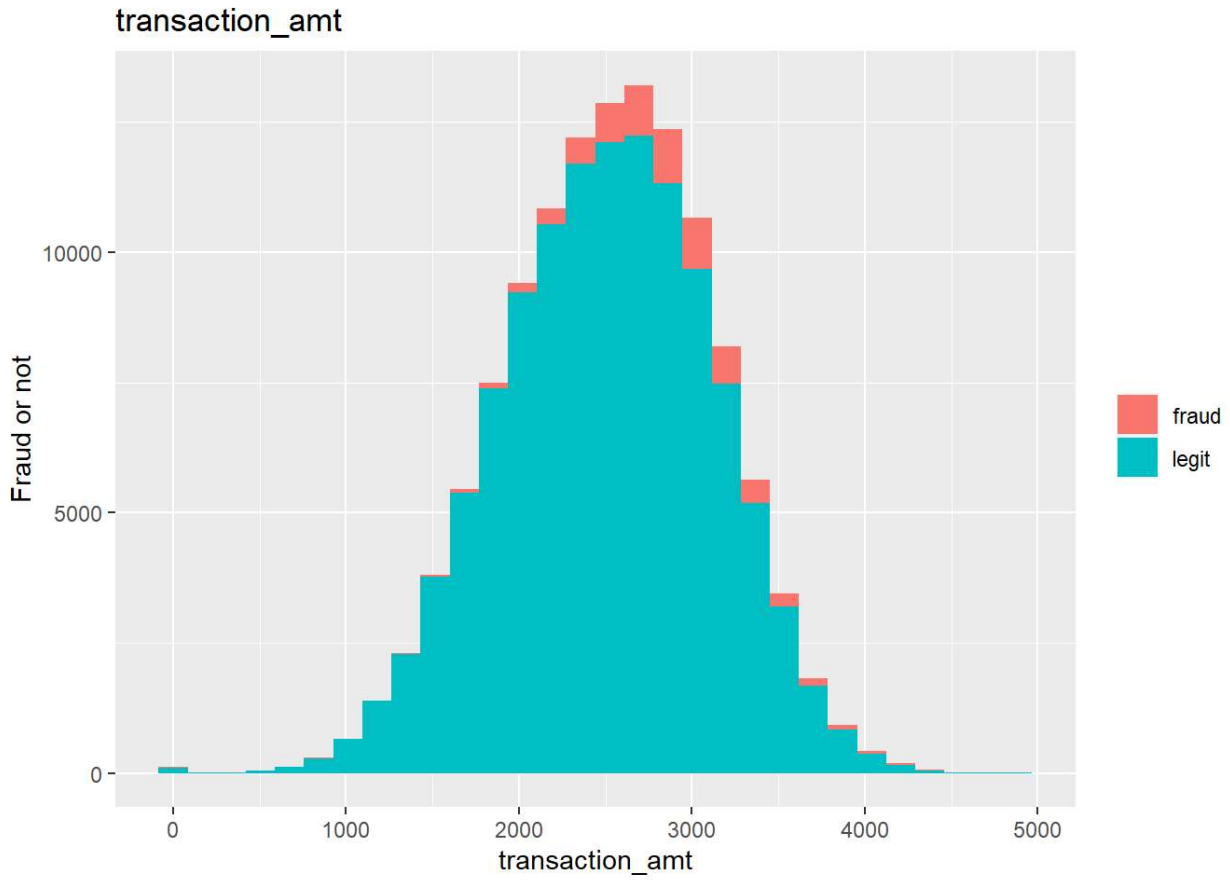

historic_velocity
1.00
0.75
0.50
0.25
0.00
Fraud or not
0
2500
5000
7500
historic_velocity
fraud
legit

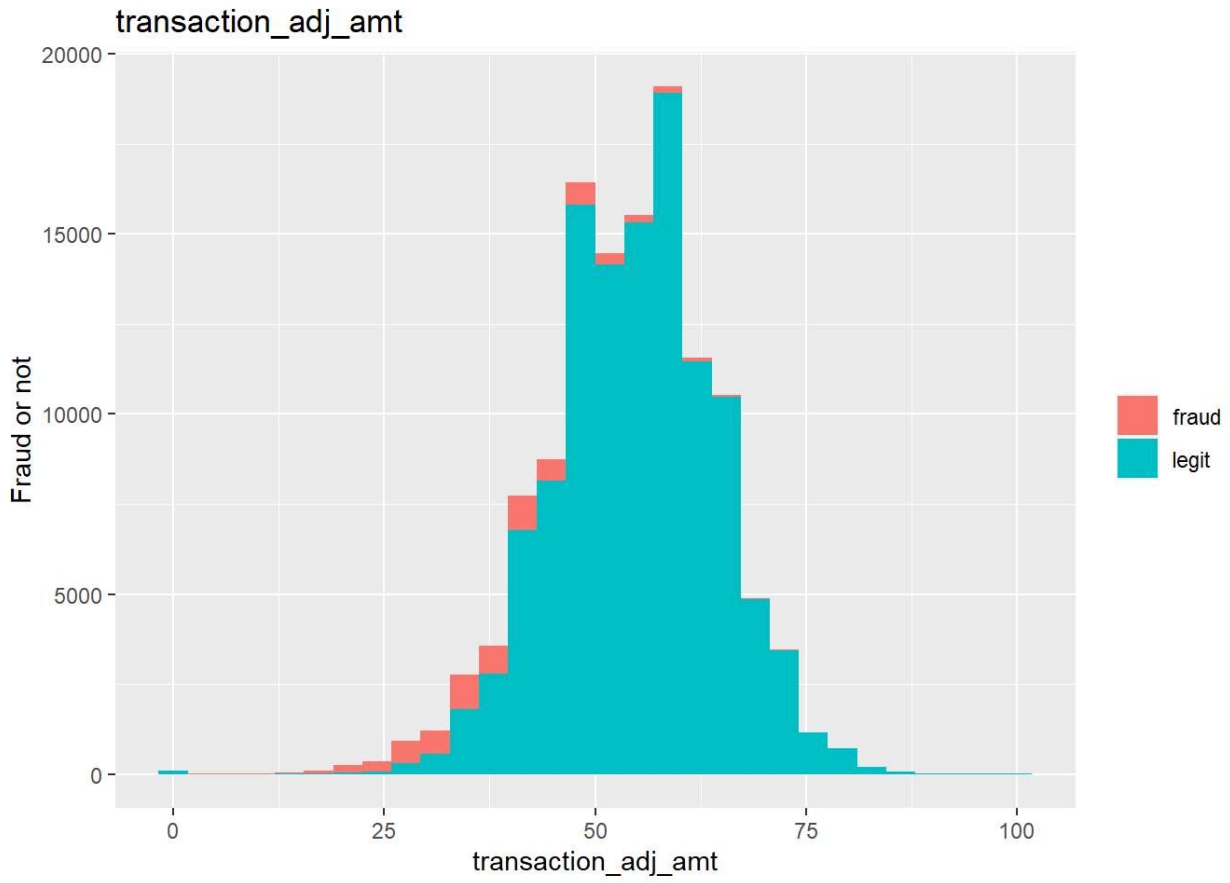


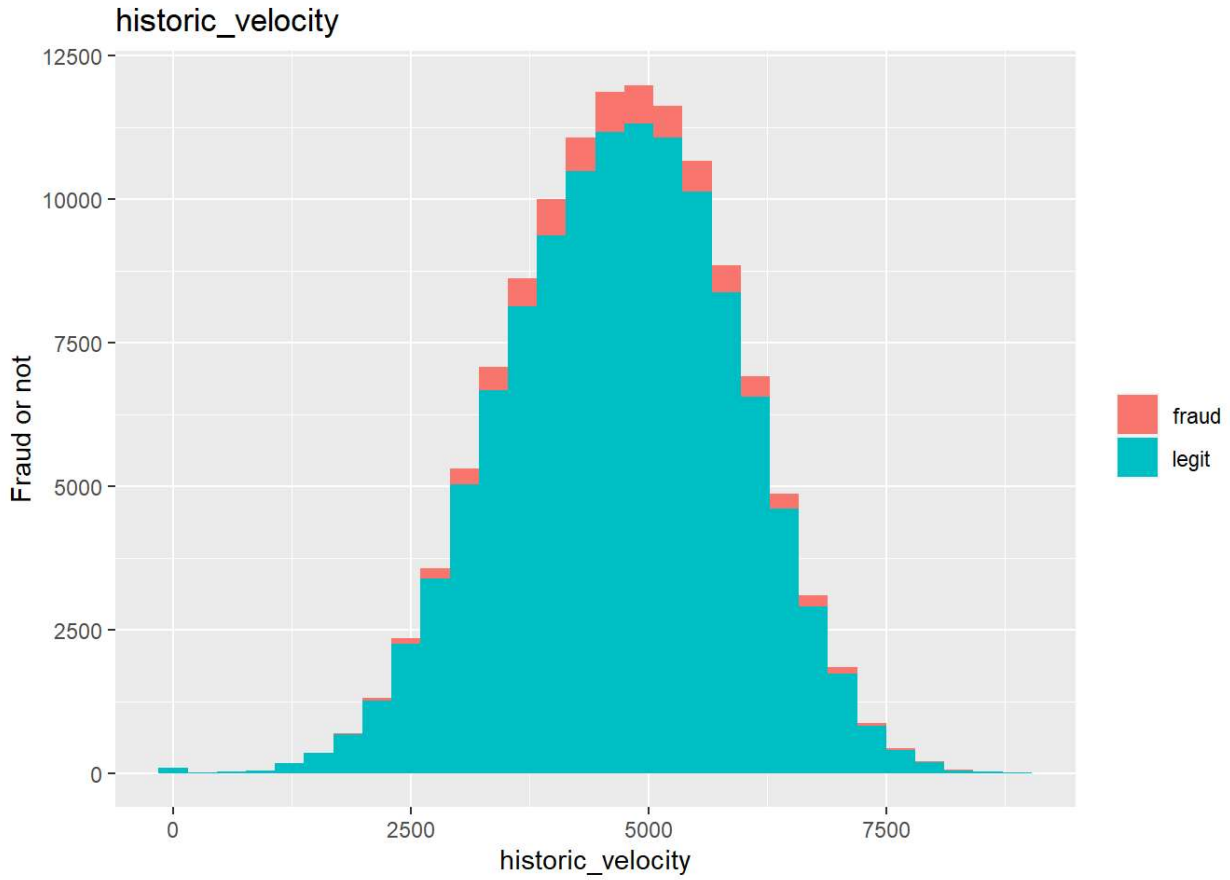


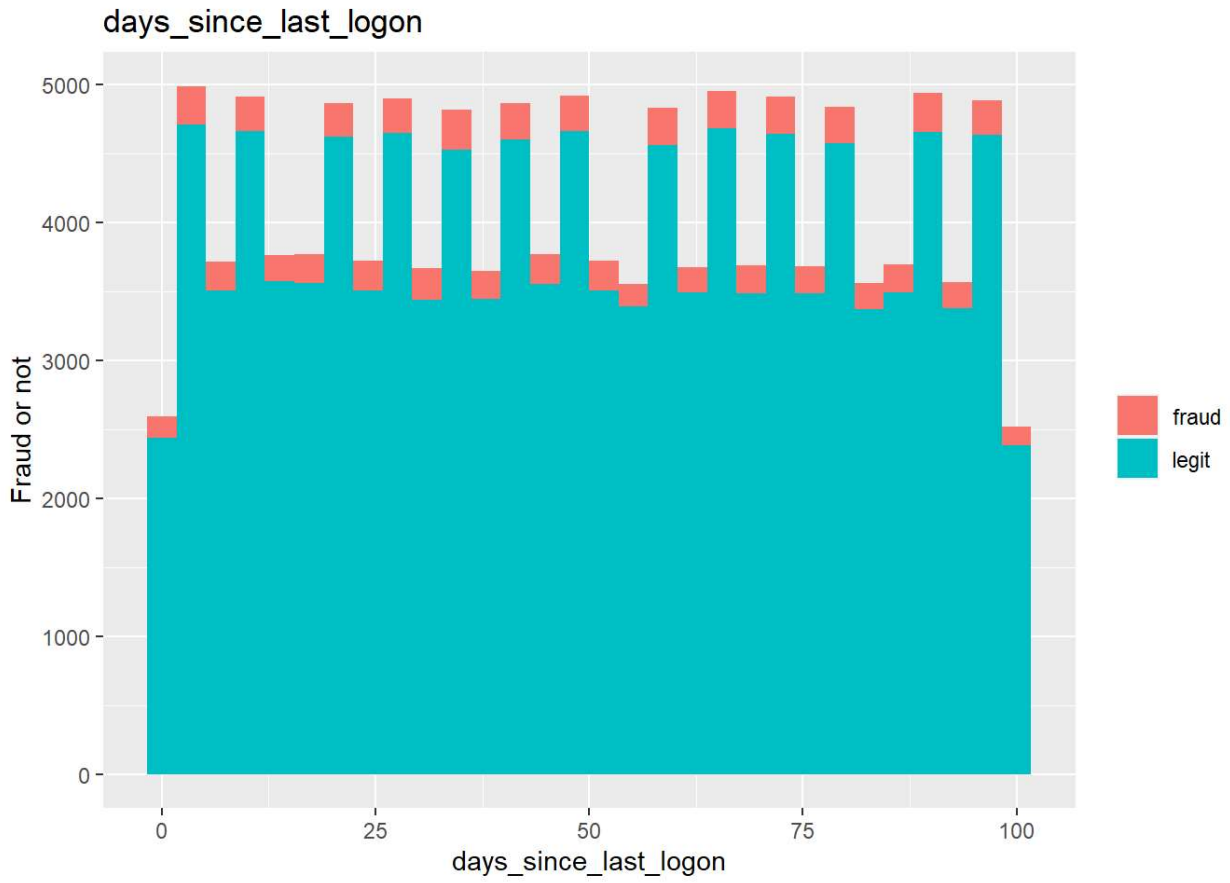

```
for (c in numerics){
    print(histogram_stack(c))
}
```

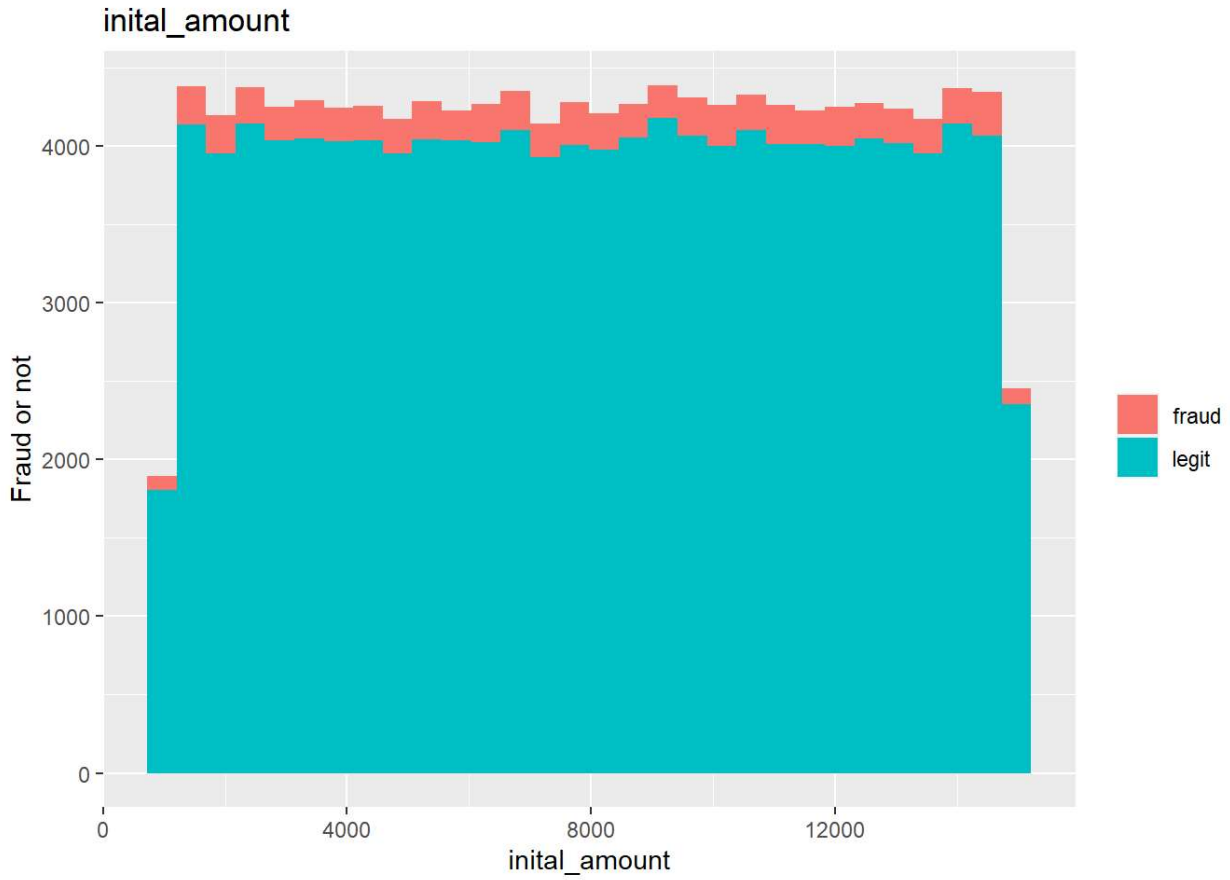

account_age_days
12000
8000
4000
0
Fraud or not
0
2500
5000
7500
account_age_days
fraud
legit


transaction_amt
10000
5000
0
Fraud or not
0
1000
2000
3000
4000
5000
transaction_amt
fraud
legit


transaction_adj_amt
Fraud or not
20000
15000
10000
5000
0
0
25
50
75
100
transaction_adj_amt
fraud
legit


historic_velocity
Fraud or not
12500
10000
7500
5000
2500
0
0
2500
5000
7500
historic_velocity
fraud
legit

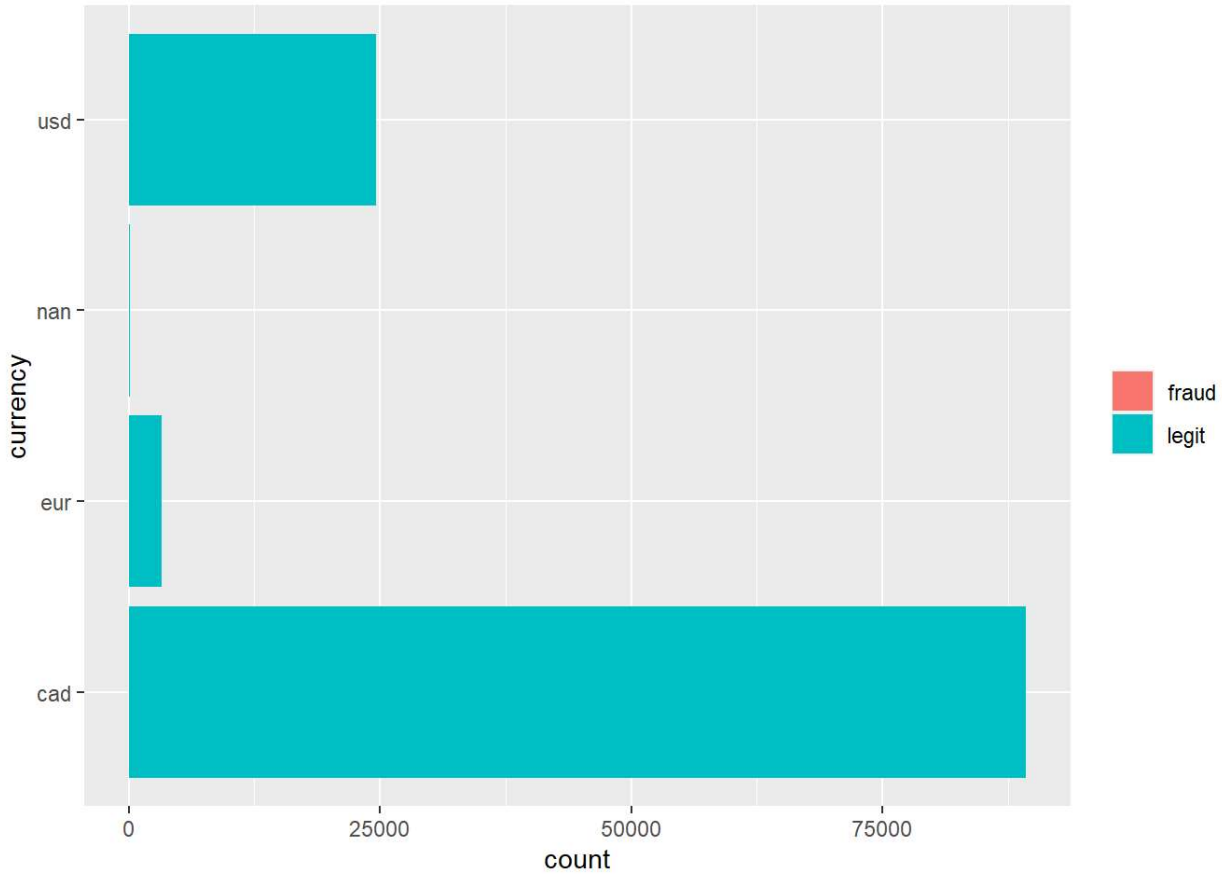


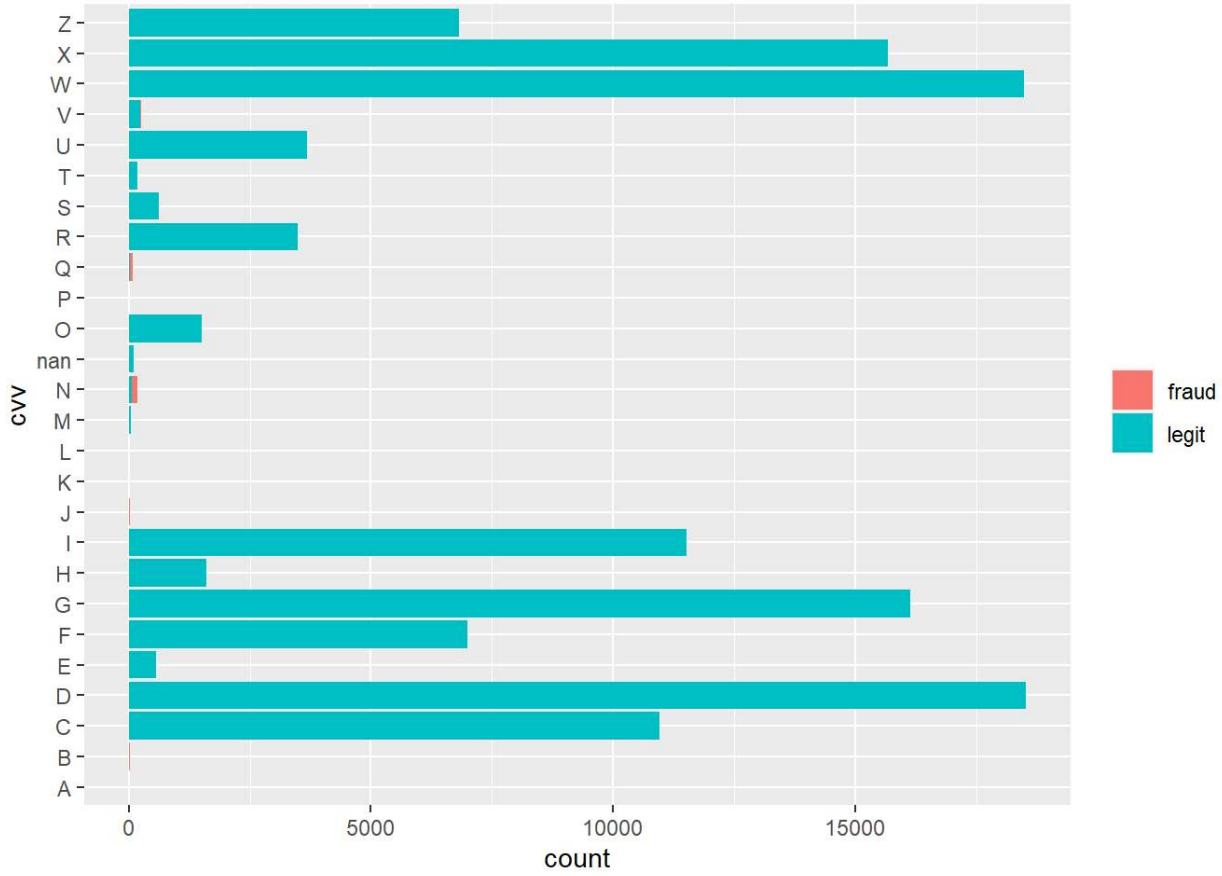


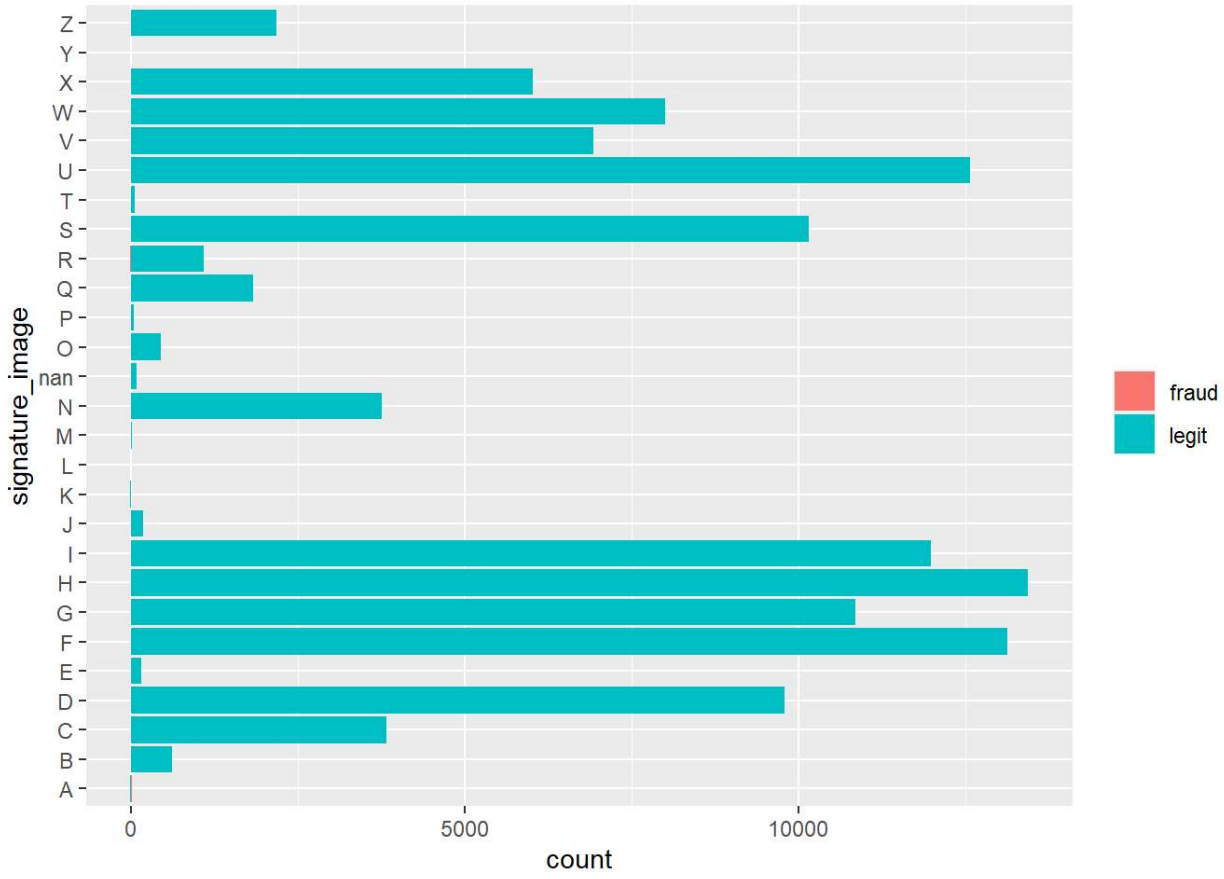
Explore character variables

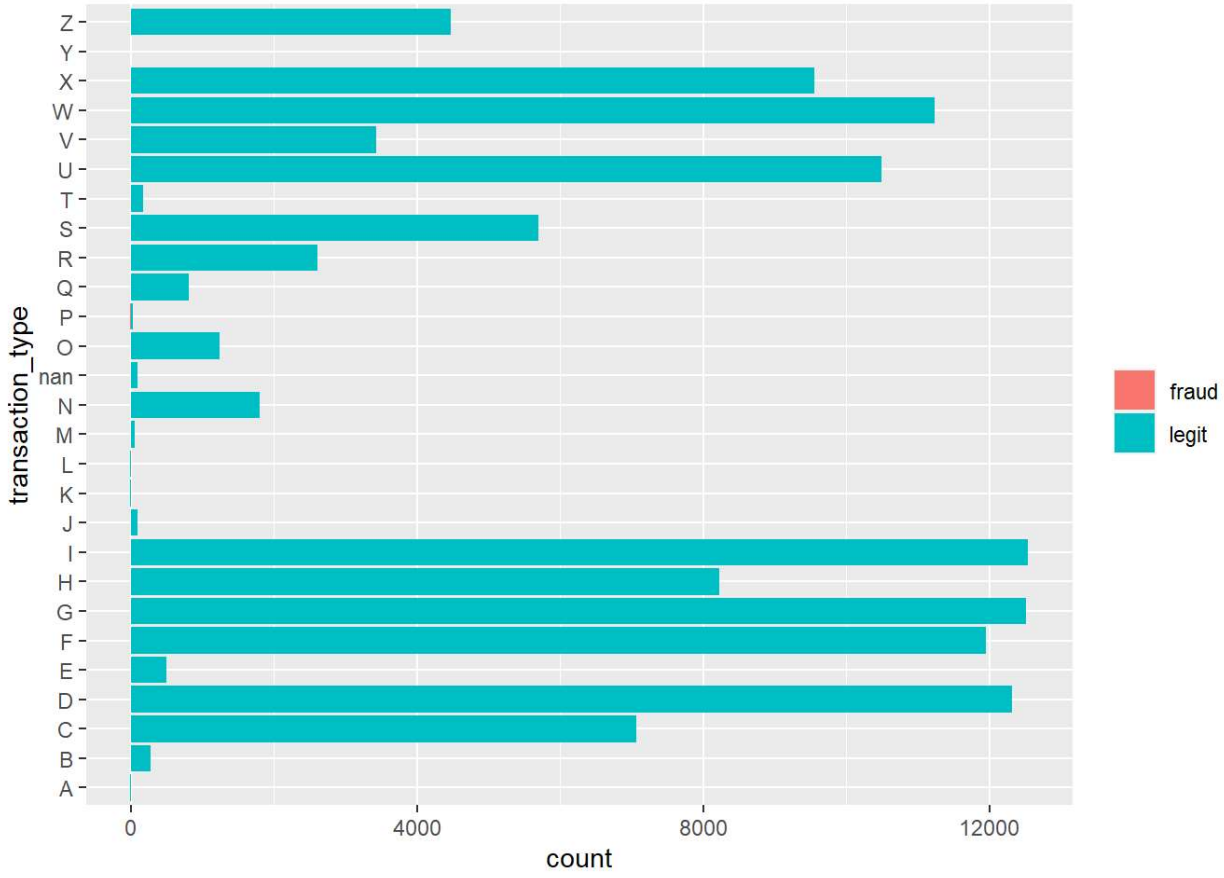
categorical variables: currency, cvv, signature_image, transaction_type, transaction_env, tranaction_initiate, billing_state

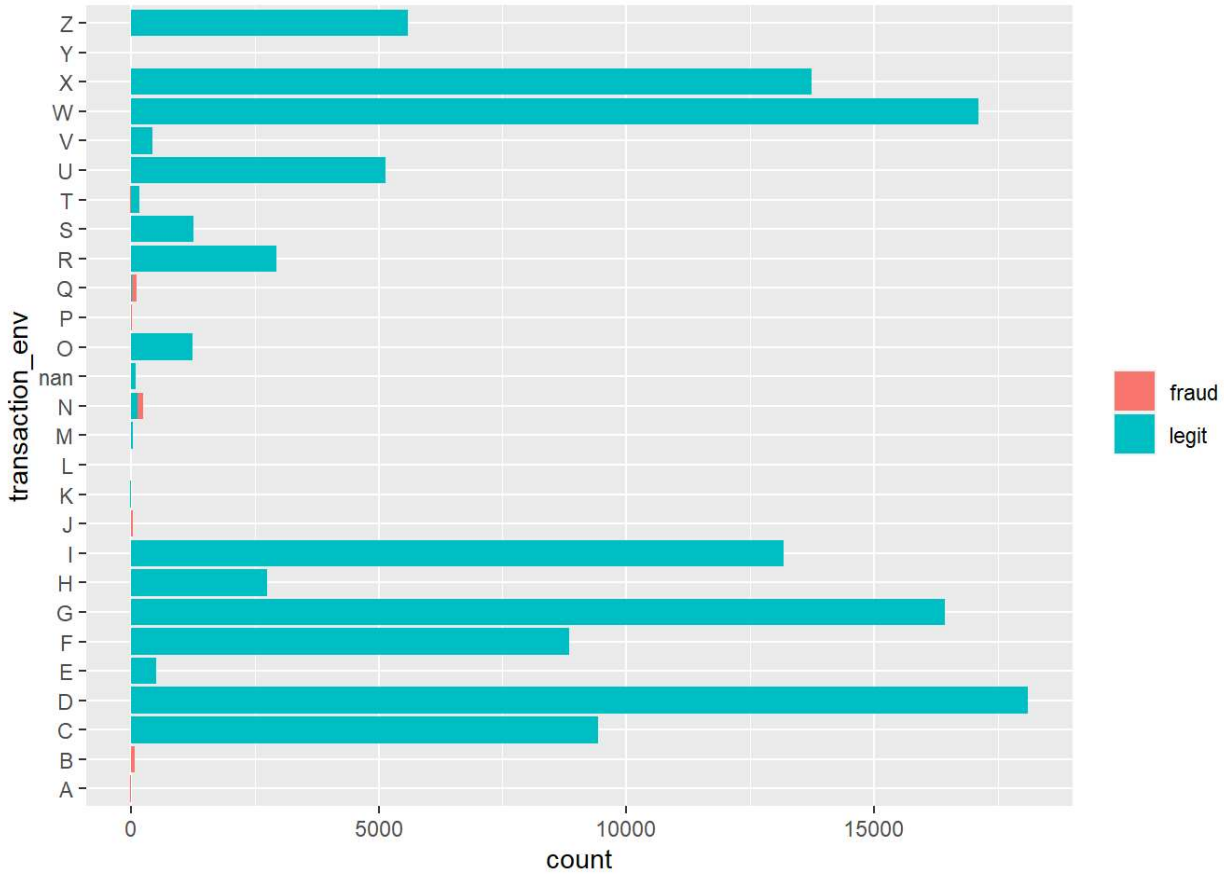
```
char_identity <- function(col){
    fraud %>%
    na.omit() %>%
    ggplot(aes(!!as.name(col),fill = as.factor(event_label))) +
    geom_bar(position = 'identity') +
    coord_flip() +
    theme(legend.title = element_blank())
}

char_fill <- function(col){
    fraud %>%
    na.omit() %>%
    ggplot(aes(!!as.name(col),fill = as.factor(event_label))) +
    geom_bar(position = 'fill') +
    coord_flip() +
    theme(legend.title = element_blank())
}

dummy <- c('currency', 'cvv', 'signature_image', 'transaction_type', 'transaction_env', 'tranaction_initiate', 'billi
          ng_state')

# -- for each character column, create a chart
for (column in dummy){
    print(char_identity(column))
}
```

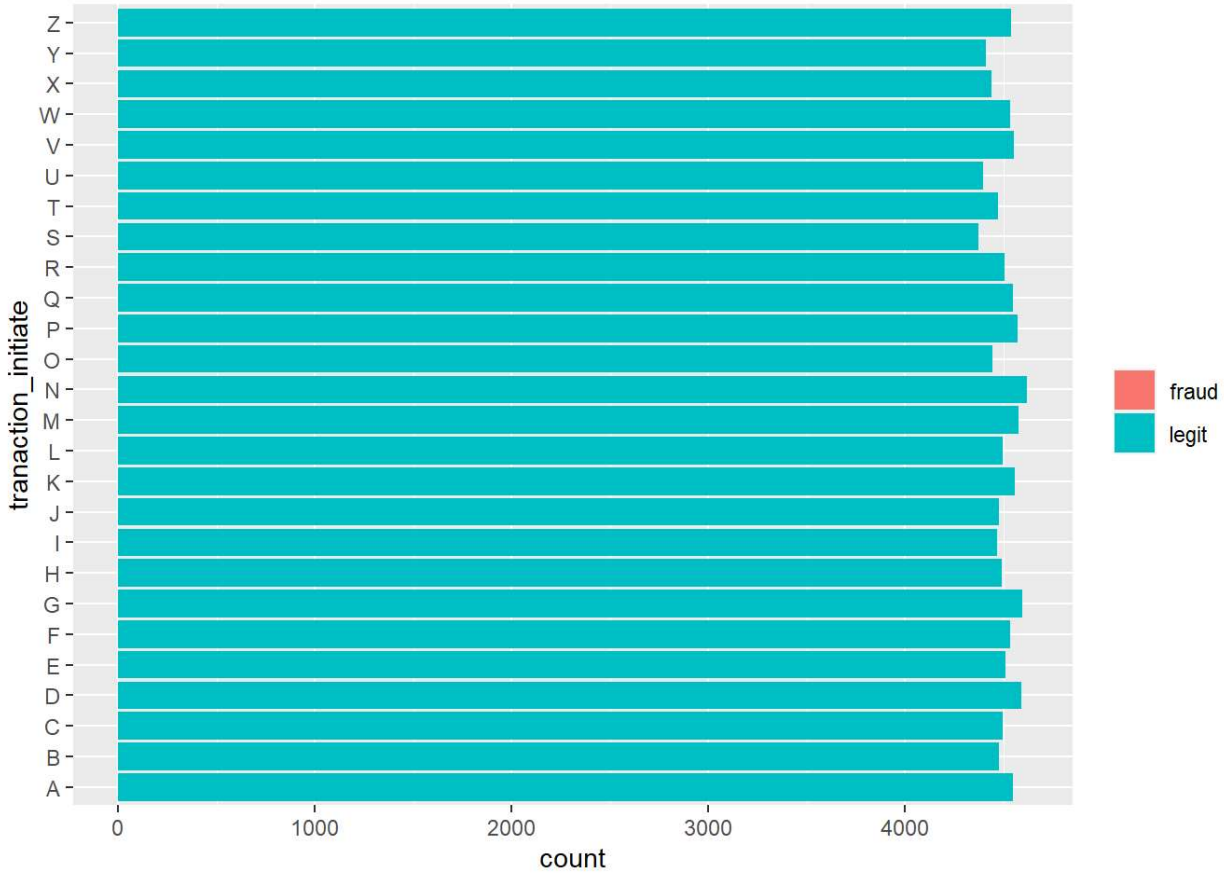

usd
nan
eur
cad
currency
0
25000
50000
75000
count
fraud
legit

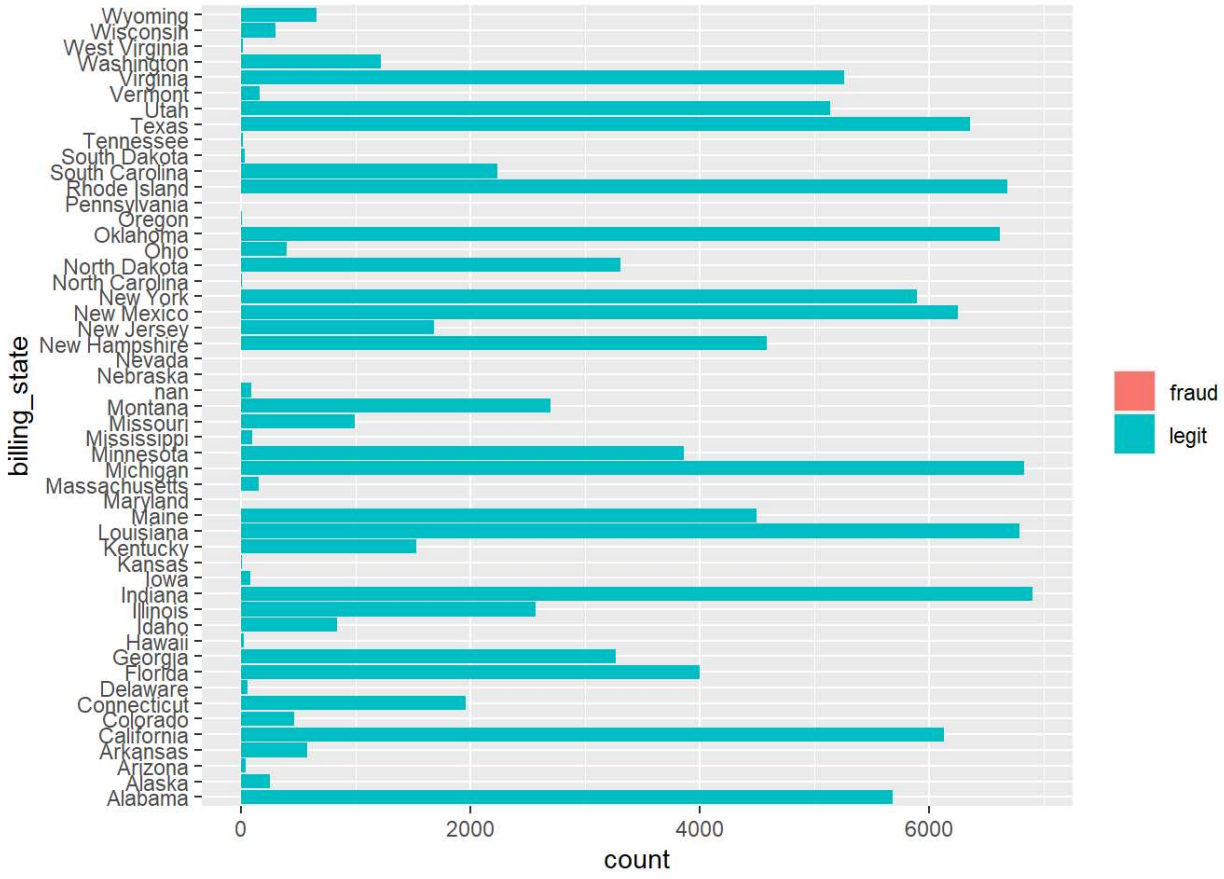

Z
X
W
V
U
T
S
R
Q
P
O
nan
N
M
L
K
J
I
H
G
F
E
D
C
B
A
cvv
0
5000
10000
15000
count
fraud
legit


signature_image
Z
Y
X
W
V
U
T
S
R
Q
P
O
nan
N
M
L
K
J
I
H
G
F
E
D
C
B
A
0
5000
10000
count
fraud
legit

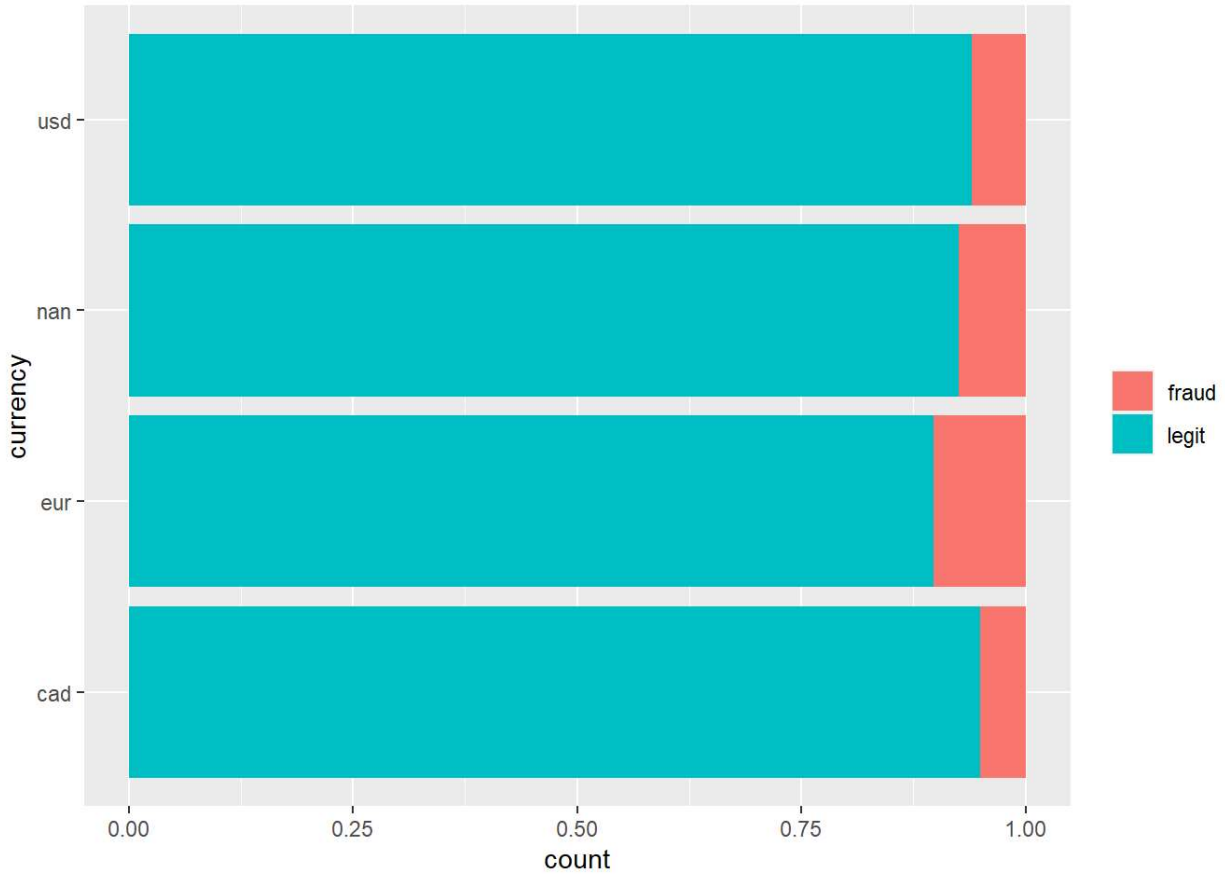

transaction_type
Z
Y
X
W
V
U
T
S
R
Q
P
O
nan
N
M
L
K
J
I
H
G
F
E
D
C
B
A
0
4000
8000
12000
count
fraud
legit

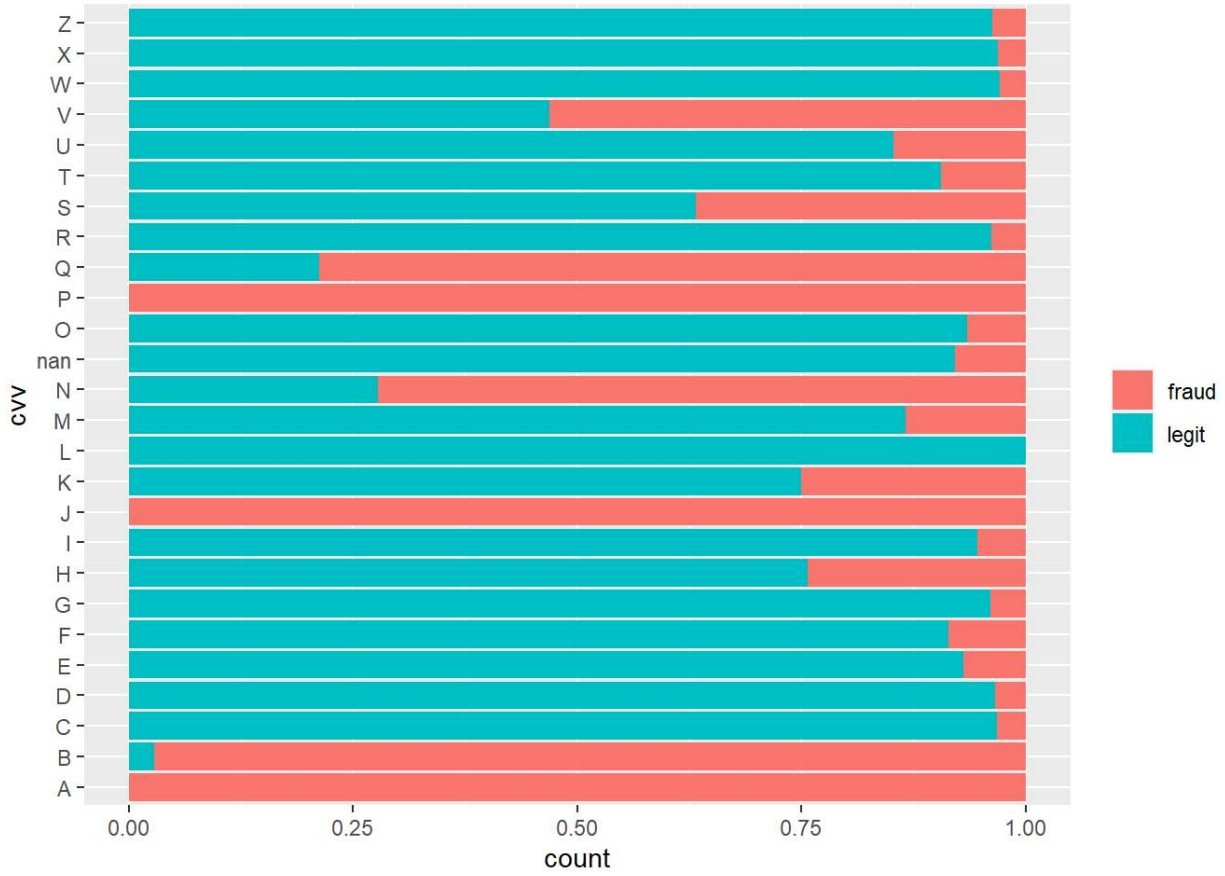

transaction_env
Z
Y
X
W
V
U
T
S
R
Q
P
O
nan
N
M
L
K
J
I
H
G
F
E
D
C
B
A
0
5000
10000
15000
count
fraud
legit

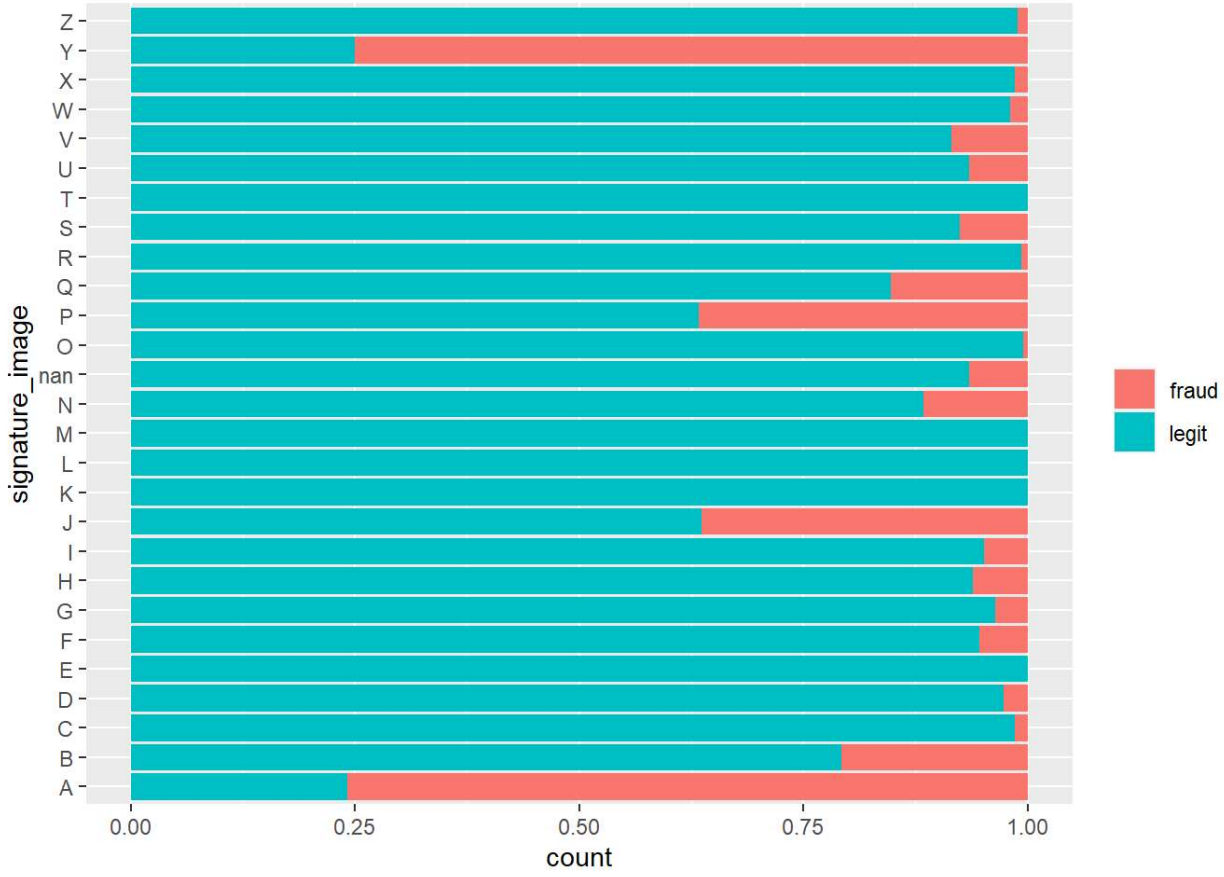

tranaction_initiate
Z
Y
X
W
V
U
T
S
R
Q
P
O
N
M
L
K
J
I
H
G
F
E
D
C
B
A
0
1000
2000
3000
4000
count
fraud
legit

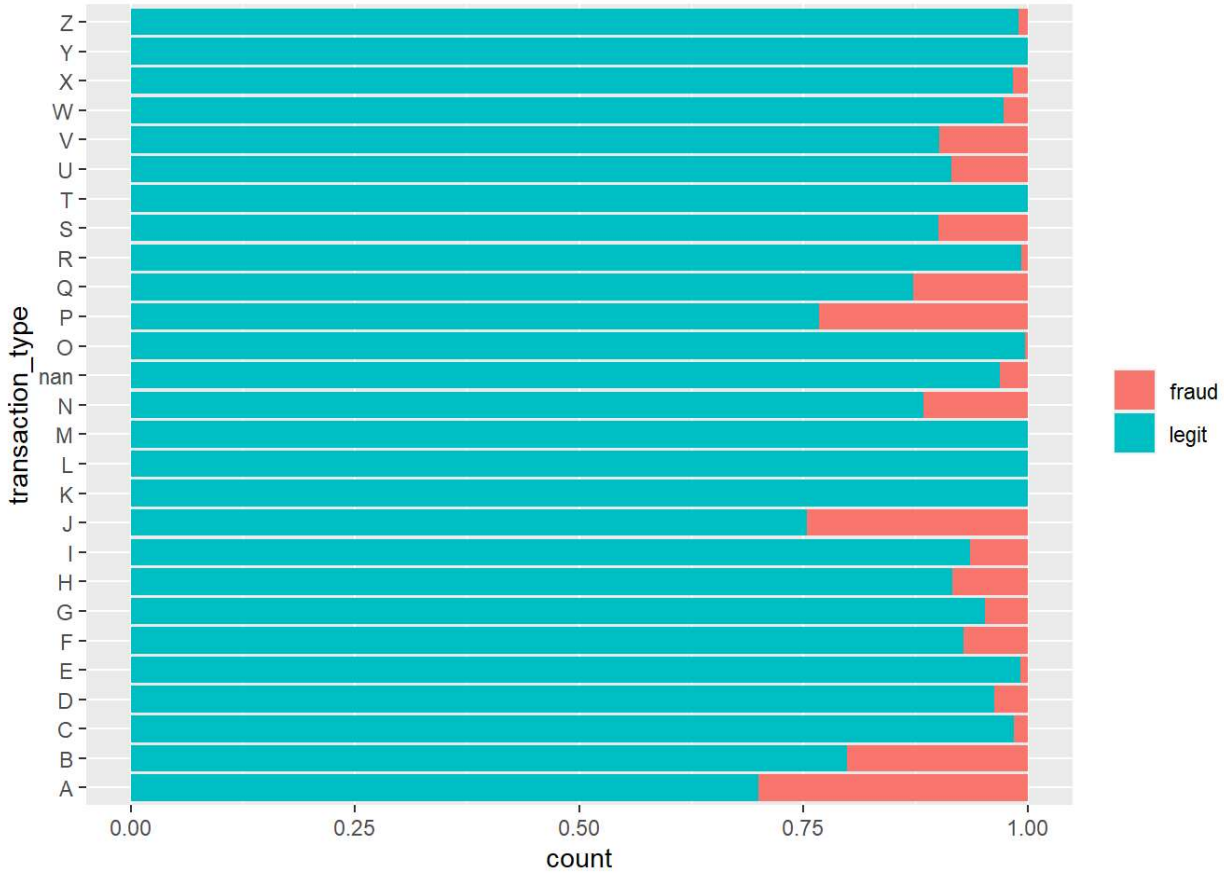


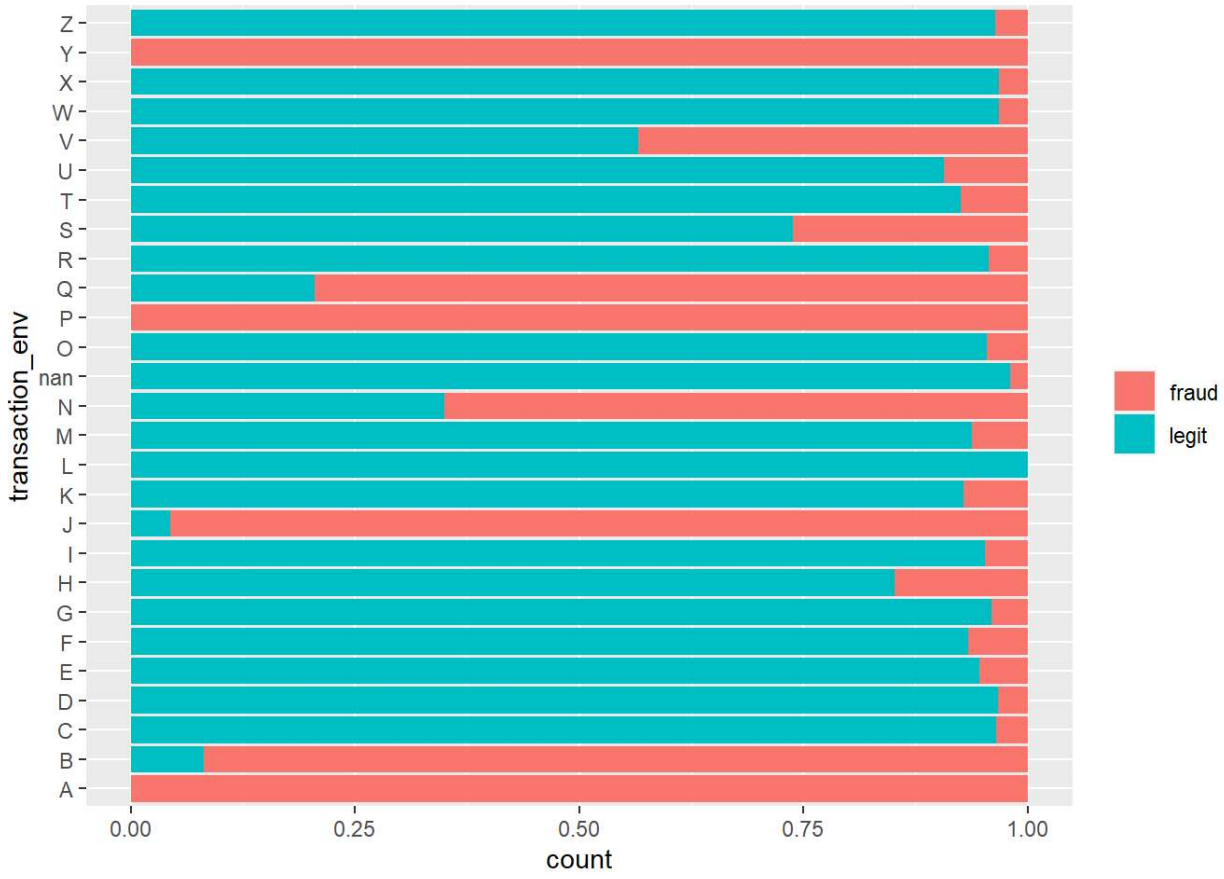

```
for (column in dummy){
    print(char_fill(column))
}
```

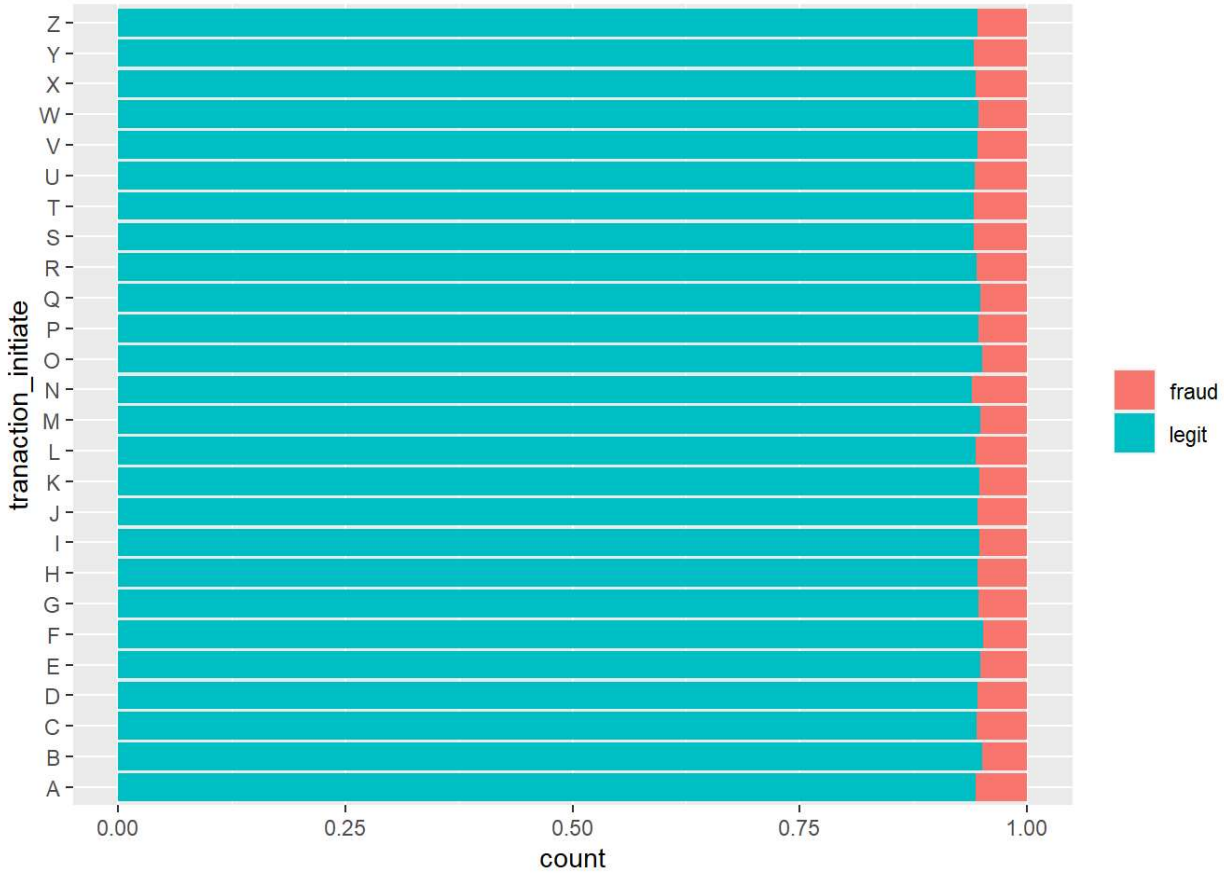

usd
nan
eur
cad
currency
0.00
0.25
0.50
0.75
1.00
count
fraud
legit

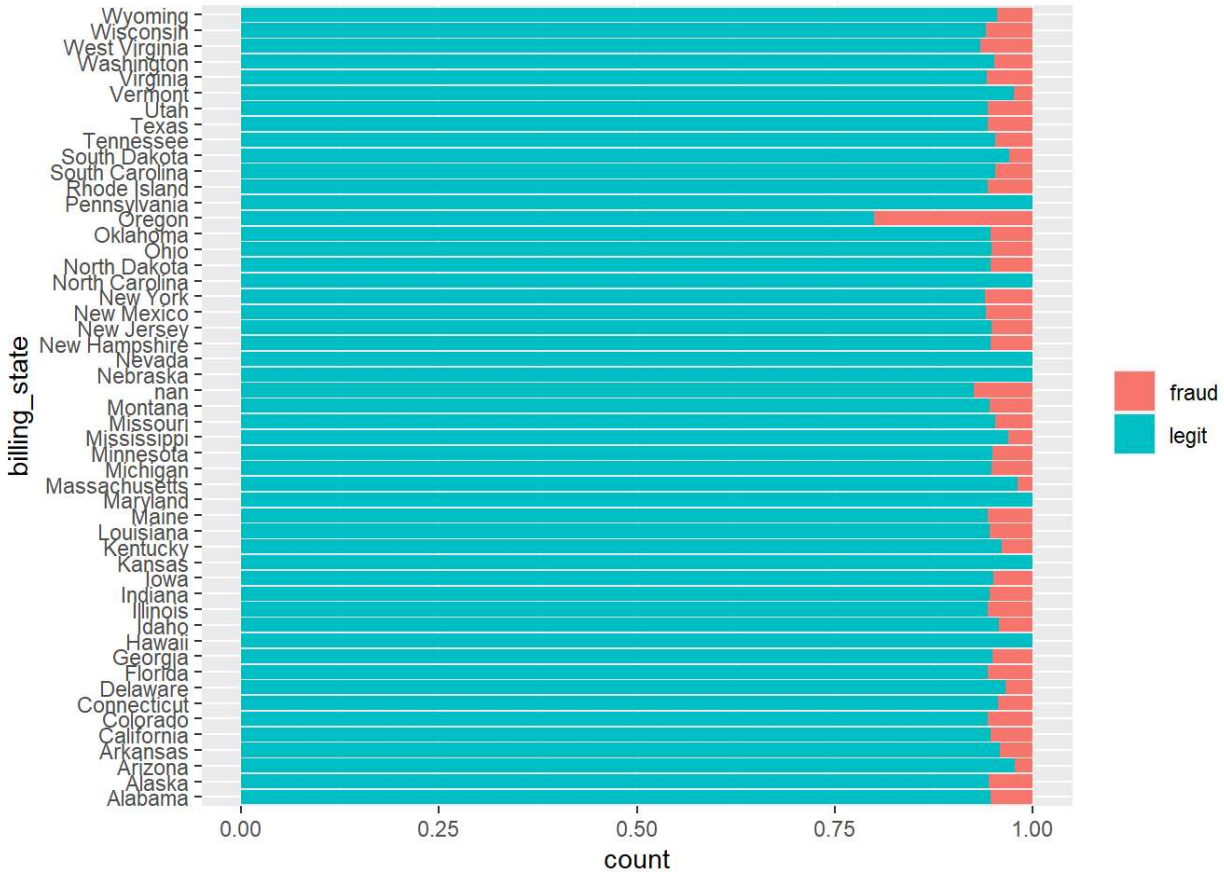

Z
X
W
V
U
T
S
R
Q
P
O
nan
N
M
L
K
J
I
H
G
F
E
D
C
B
A
cvv
0.00
0.25
0.50
0.75
1.00
count
fraud
legit


signature_image
Z
Y
X
W
V
U
T
S
R
Q
P
O
nan
N
M
L
K
J
I
H
G
F
E
D
C
B
A
0.00
0.25
0.50
0.75
1.00
count
fraud
legit


transaction_type
Z
Y
X
W
V
U
T
S
R
Q
P
O
nan
N
M
L
K
J
I
H
G
F
E
D
C
B
A
0.00
0.25
0.50
0.75
1.00
count
fraud
legit


transaction_env
Z
Y
X
W
V
U
T
S
R
Q
P
O
nan
N
M
L
K
J
I
H
G
F
E
D
C
B
A
0.00
0.25
0.50
0.75
1.00
count
fraud
legit


tranaction_initiate
Z
Y
X
W
V
U
T
S
R
Q
P
O
N
M
L
K
J
I
H
G
F
E
D
C
B
A
0.00
0.25
0.50
0.75
1.00
count
fraud
legit



4. Transform

Convert categories to factors

```
fraud <- fraud %>%
    mutate(event_label = as.factor(event_label)) %>%
    mutate_if(is.character,factor)

fraud_kaggle <- fraud_kaggle %>%
    mutate_if(is.character,factor)

data <- fraud %>%
  select(account_age_days, transaction_amt, transaction_adj_amt, historic_velocity, currency, cvv, signature_image, t
         ransaction_type, transaction_env, billing_state, event_label)

table(data$event_label)
```

```
##
##  fraud  legit
##   6785 118215
```

5. Partition your data into 70/30 train/test split

```
set.seed(43)

# -- performs our train / test split
split <- initial_split(data, prop = 0.7)

# -- extract the training data form our banana split
train <- training(split)
# -- extract the test data
test <- testing(split)

sprintf("Train PCT : %1.2f%%", nrow(train)/ nrow(data) * 100)
```

```
## [1] "Train PCT : 70.00%"
```

```
sprintf("Test  PCT : %1.2f%%", nrow(test)/ nrow(data) * 100)
```

```
## [1] "Test  PCT : 30.00%"
```

6. Define Recipe

```
model_recipe <- recipe(event_label ~ ., data = data) %>%
    step_impute_median(all_numeric_predictors()) %>%
    step_unknown(all_nominal_predictors()) %>%
    step_scale(all_numeric_predictors()) %>%
    step_dummy(all_nominal_predictors()) %>%
    step_nzv(all_predictors())

bake(model_recipe %>% prep(), train, composition = "tibble") %>% head()
```

account_age_days <dbl>	transaction_amt <dbl>	transaction_adj_amt <dbl>	historic_velocity <dbl>	event_label <fct>	currency_usd <int>	c <…
4.548136	3.336608	5.209477	4.202267	legit	0	
3.548924	4.104700	4.619725	2.660849	legit	0	
4.131223	3.208592	4.914601	3.410208	legit	0	
2.714238	3.279165	5.602645	5.097310	legit	0	
5.220881	4.613480	6.094105	5.644048	legit	1	
5.485327	4.116189	6.094105	5.147546	legit	0	

6 rows | 1-8 of 49 columns

```
bake_train <- bake(model_recipe %>% prep(), train, composition = "tibble")
bake_test <- bake(model_recipe %>% prep(), test, composition = "tibble")



model_recipe_tree <- recipe(event_label ~ ., data = data) %>%
    step_impute_median(all_numeric_predictors()) %>%
    step_unknown(all_nominal_predictors()) %>%
    step_dummy(all_nominal_predictors())

bake(model_recipe_tree %>% prep(), train, composition = "tibble") %>% head()
```

account_age_days <dbl>	transaction_amt <dbl>	transaction_adj_amt <dbl>	historic_velocity <dbl>	event_label <fct>	currency_eur <int>
5280	2033	53	5019	legit	0
4120	2501	47	3178	legit	0
4796	1955	50	4073	legit	0
3151	1998	57	6088	legit	0
6061	2811	62	6741	legit	0
6368	2508	62	6148	legit	0

6 rows | 1-7 of 167 columns

```
bake_train_tree <- bake(model_recipe %>% prep(), train, composition = "tibble")
bake_test_tree <- bake(model_recipe %>% prep(), test, composition = "tibble")
```

7. Define your Model(s)

```
rf_model_1 <- rand_forest(trees=10, min_n = 10) %>%
  set_mode("classification") %>%
  set_engine("ranger", importance="impurity")

rf_model_2 <- rand_forest(trees=10, min_n = 10) %>%
  set_mode("classification") %>%
  set_engine("ranger", importance="permutation")

rf_model_3 <- rand_forest(trees=1200, min_n = 10) %>%
  set_mode("classification") %>%
  set_engine("ranger", importance="impurity")
```

8. Workflow

```
tree_workflow_1 <- workflow() %>%
  add_recipe(model_recipe_tree) %>%
  add_model(rf_model_1) %>%
  fit(train)

tree_workflow_2 <- workflow() %>%
  add_recipe(model_recipe_tree) %>%
  add_model(rf_model_2) %>%
  fit(train)

tree_workflow_3 <- workflow() %>%
  add_recipe(model_recipe_tree) %>%
  add_model(rf_model_3) %>%
  fit(train)
```

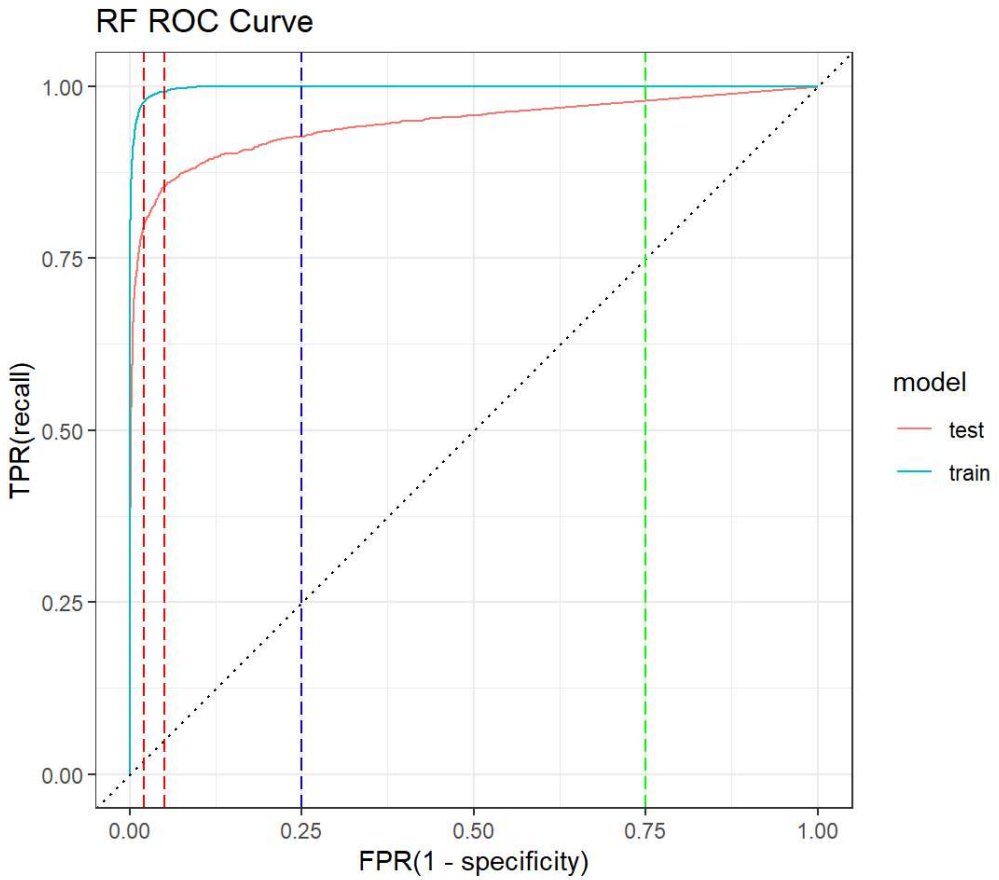
9. Evaluation (rf_model_1)

```
options(yardstick.event_first = TRUE)
  # -- score training
  predict(tree_workflow_1, train, type="prob") %>%
    bind_cols(predict(tree_workflow_1, train, type="class")) %>%
    bind_cols(., train)-> scored_train_1

  # -- score testing
  predict(tree_workflow_1, test, type="prob") %>%
      bind_cols(predict(tree_workflow_1, test, type="class")) %>%
      bind_cols(., test) -> scored_test_1
  # -- Metrics (AUC / Accuracy)
  scored_train_1 %>%
    metrics(event_label, .pred_fraud, estimate = .pred_class) %>%
    mutate(part="training") %>%
    bind_rows(scored_test_1 %>%
                metrics(event_label, .pred_fraud, estimate = .pred_class) %>%
                mutate(part="testing") ) %>%
    filter(.metric %in% c('accuracy','roc_auc')) %>%
    pivot_wider(names_from = .metric, values_from=.estimate)
```

.estimator	part	accuracy	roc_auc
<chr>	<chr>	<dbl>	<dbl>
binary	training	0.9866971	0.9978394
binary	testing	0.9746400	0.9442435

2 rows

```
# -- ROC Charts
scored_train_1 %>%
  mutate(model = "train") %>%
  bind_rows(scored_test_1 %>%
              mutate(model="test")) %>%
  group_by(model) %>%
  roc_curve(event_label, .pred_fraud) %>%
  autoplot() +
  geom_vline(xintercept = 0.0213,
             color = "red",
             linetype = "longdash") +
  geom_vline(xintercept = 0.05, # 5% FPR
             color = "red",
             linetype = "longdash") +
  geom_vline(xintercept = 0.25,    # 25% FPR
             color = "blue",
             linetype = "longdash") +
  geom_vline(xintercept = 0.75,    # 75% FPR
             color = "green",
             linetype = "longdash") +
  labs(title = "RF ROC Curve" , x = "FPR(1 - specificity)", y = "TPR(recall)")
```

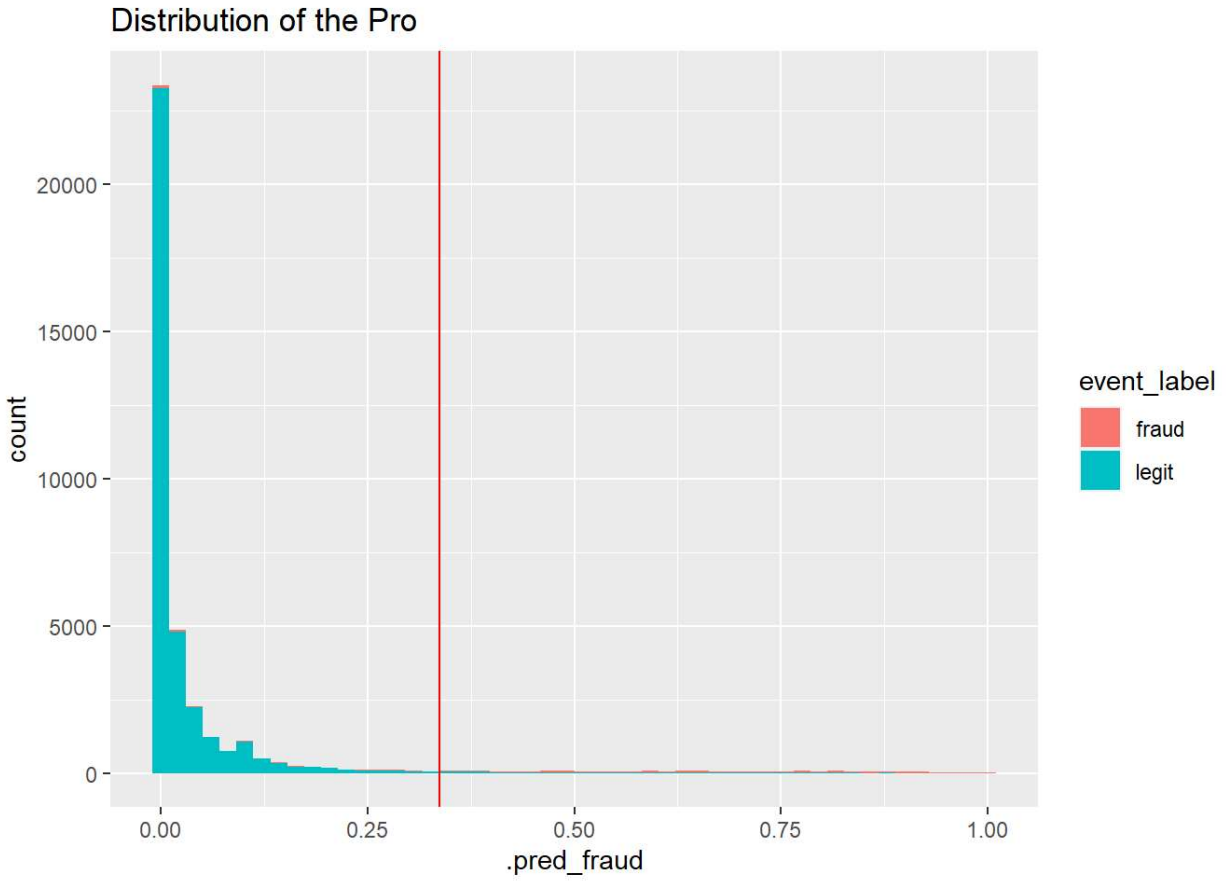


```
scored_train_1 %>%
  mutate(model = "train") %>%
  bind_rows(scored_test_1 %>%
              mutate(model="test")) %>%
  group_by(model) %>%
  spec(event_label, .pred_class) %>%
  mutate(fpr=1- .estimate)
```

model	.metric	.estimator	.estimate	fpr
<chr>	<chr>	<chr>	<dbl>	<dbl>
test	spec	binary	0.9964508	0.0035491958
train	spec	binary	0.9993230	0.0006770317

2 rows

```
# histogram of probability of fraud
scored_test_1 %>%
  ggplot(aes(.pred_fraud, fill = event_label)) +
  geom_histogram(bins = 50) +
  geom_vline(xintercept = 0.337, color = "red") +
  labs(title = paste("Distribution of the Pro"))
```



```
scored_train_1 %>%
  precision(event_label, .pred_class) %>%
  mutate(part="training") %>%
  bind_rows(
  scored_test_1 %>%
  precision(event_label,.pred_class) %>%
    mutate(part="testing"))
```

.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
precision	binary	0.9850027	training
precision	binary	0.9030769	testing

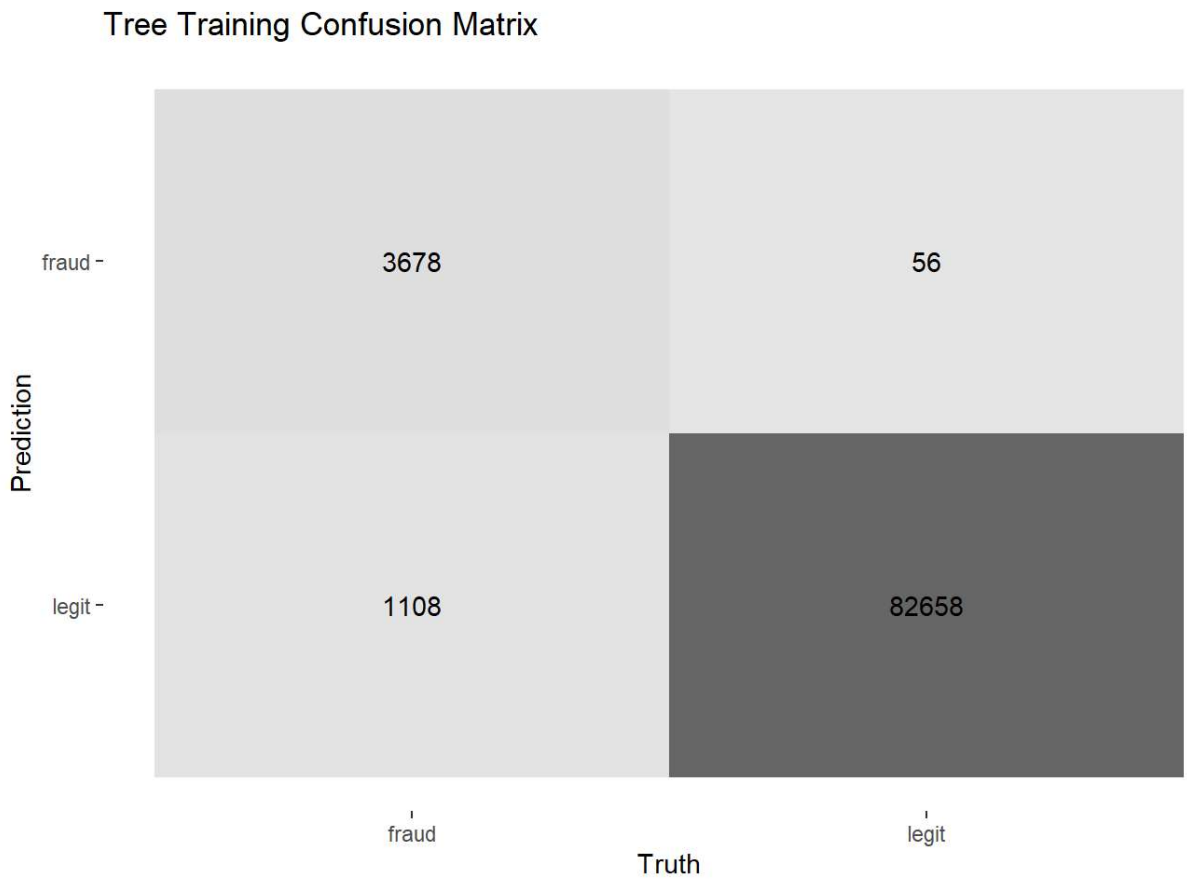
2 rows

```
scored_train_1 %>%
  recall(event_label, .pred_class) %>%
  mutate(part="training") %>%
  bind_rows(
  scored_test_1 %>%
  recall(event_label,.pred_class) %>%
    mutate(part="testing")
  )
```

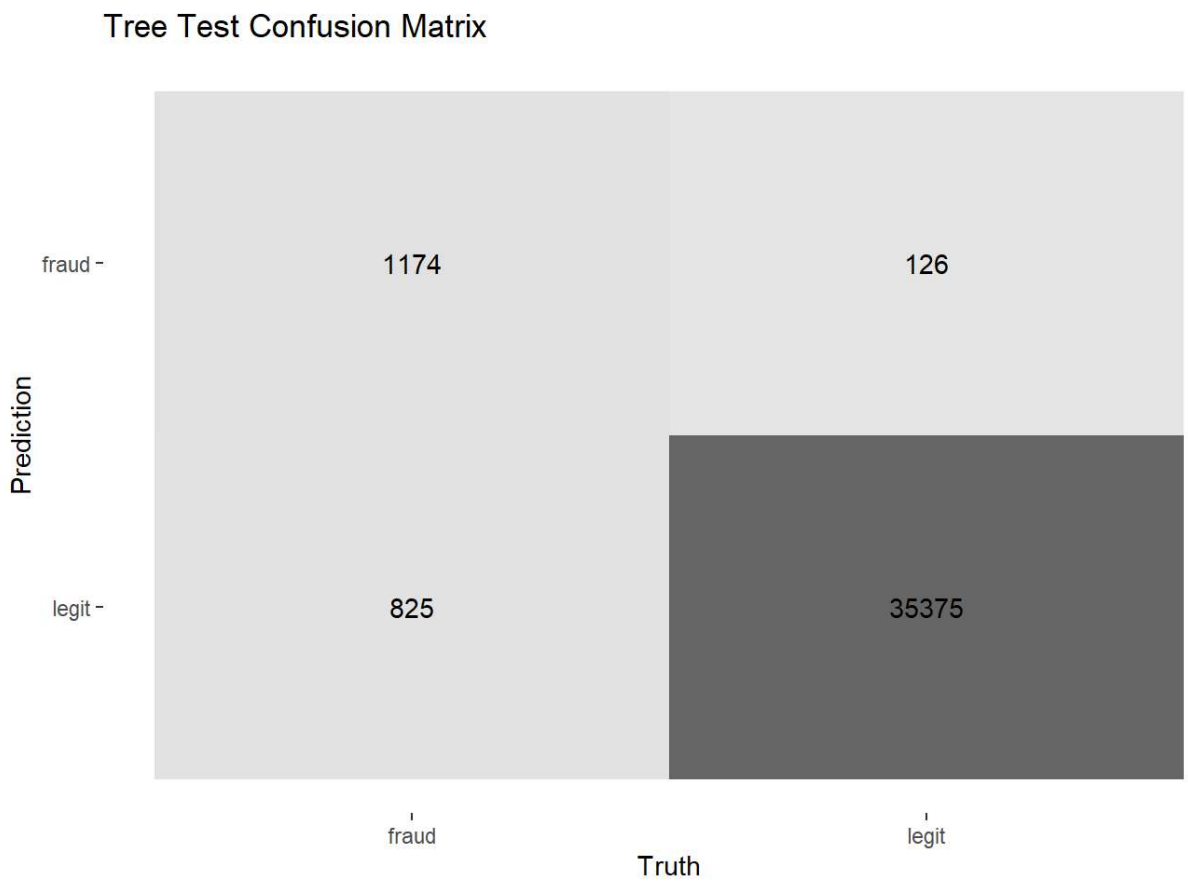
.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
recall	binary	0.7684914	training
recall	binary	0.5872936	testing

2 rows

```
scored_train_1 %>%
  conf_mat(
  truth = event_label,
  estimate = .pred_class,
  dnn = c("Prediction", "Truth")
) %>%
  autoplot(type = "heatmap") +
  labs(title="Tree Training Confusion Matrix")
```




```
scored_test_1 %>%
  conf_mat(
  truth = event_label,
  estimate = .pred_class,
  dnn = c("Prediction", "Truth")
) %>%
  autoplot(type = "heatmap") +
  labs(title="Tree Test Confusion Matrix")
```



Evaluation (rf_model_2)

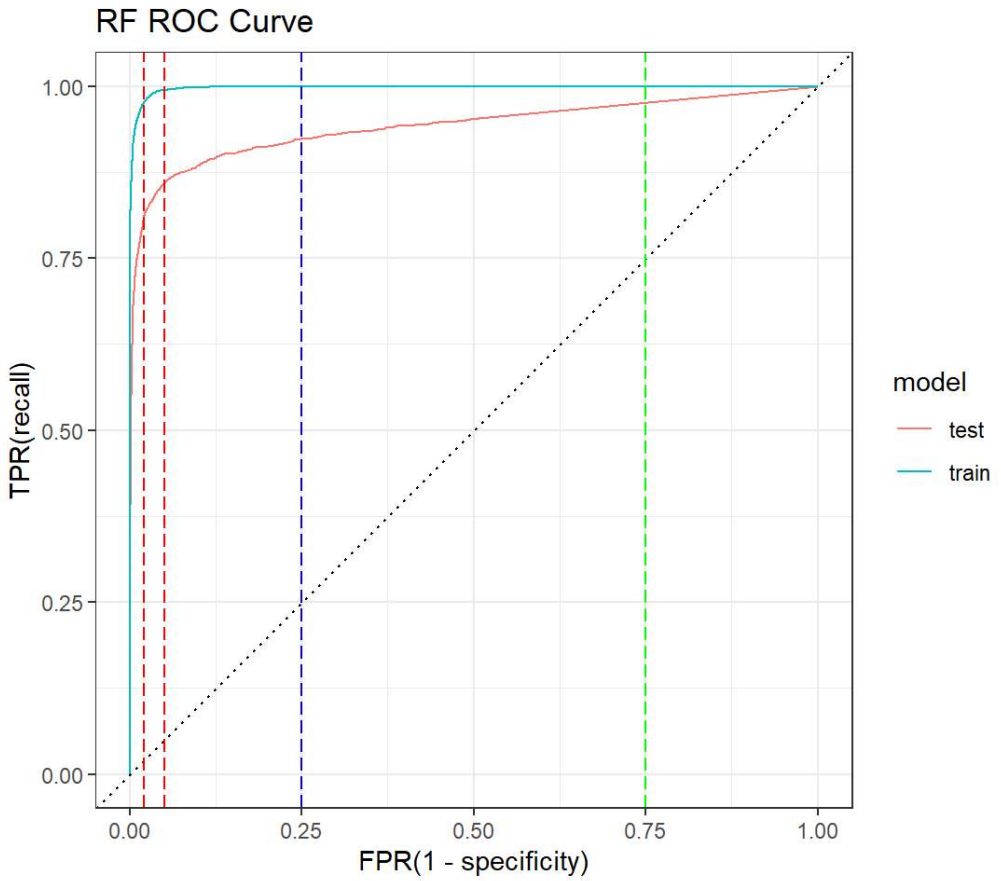
```
options(yardstick.event_first = TRUE)
  # -- score training
  predict(tree_workflow_2, train, type="prob") %>%
    bind_cols(predict(tree_workflow_2, train, type="class")) %>%
    bind_cols(., train)-> scored_train_2

  # -- score testing
  predict(tree_workflow_2, test, type="prob") %>%
      bind_cols(predict(tree_workflow_2, test, type="class")) %>%
      bind_cols(., test) -> scored_test_2
  # -- Metrics (AUC / Accuracy)
  scored_train_2 %>%
    metrics(event_label, .pred_fraud, estimate = .pred_class) %>%
    mutate(part="training") %>%
    bind_rows(scored_test_2 %>%
                metrics(event_label, .pred_fraud, estimate = .pred_class) %>%
                mutate(part="testing") ) %>%
    filter(.metric %in% c('accuracy','roc_auc')) %>%
    pivot_wider(names_from = .metric, values_from=.estimate)
```

.estimator	part	accuracy	roc_auc
<chr>	<chr>	<dbl>	<dbl>
binary	training	0.9876229	0.9979603

.estimator	part	accuracy	roc_auc
<chr>	<chr>	<dbl>	<dbl>
binary	testing	0.9755467	0.9414496

2 rows

```
# -- ROC Charts
scored_train_2 %>%
  mutate(model = "train") %>%
  bind_rows(scored_test_2 %>%
              mutate(model="test")) %>%
  group_by(model) %>%
  roc_curve(event_label, .pred_fraud) %>%
  autoplot() +
  geom_vline(xintercept = 0.0213,
             color = "red",
             linetype = "longdash") +
  geom_vline(xintercept = 0.05, # 5% FPR
             color = "red",
             linetype = "longdash") +
  geom_vline(xintercept = 0.25,    # 25% FPR
             color = "blue",
             linetype = "longdash") +
  geom_vline(xintercept = 0.75,    # 75% FPR
             color = "green",
             linetype = "longdash") +
  labs(title = "RF ROC Curve" , x = "FPR(1 - specificity)", y = "TPR(recall)")
```

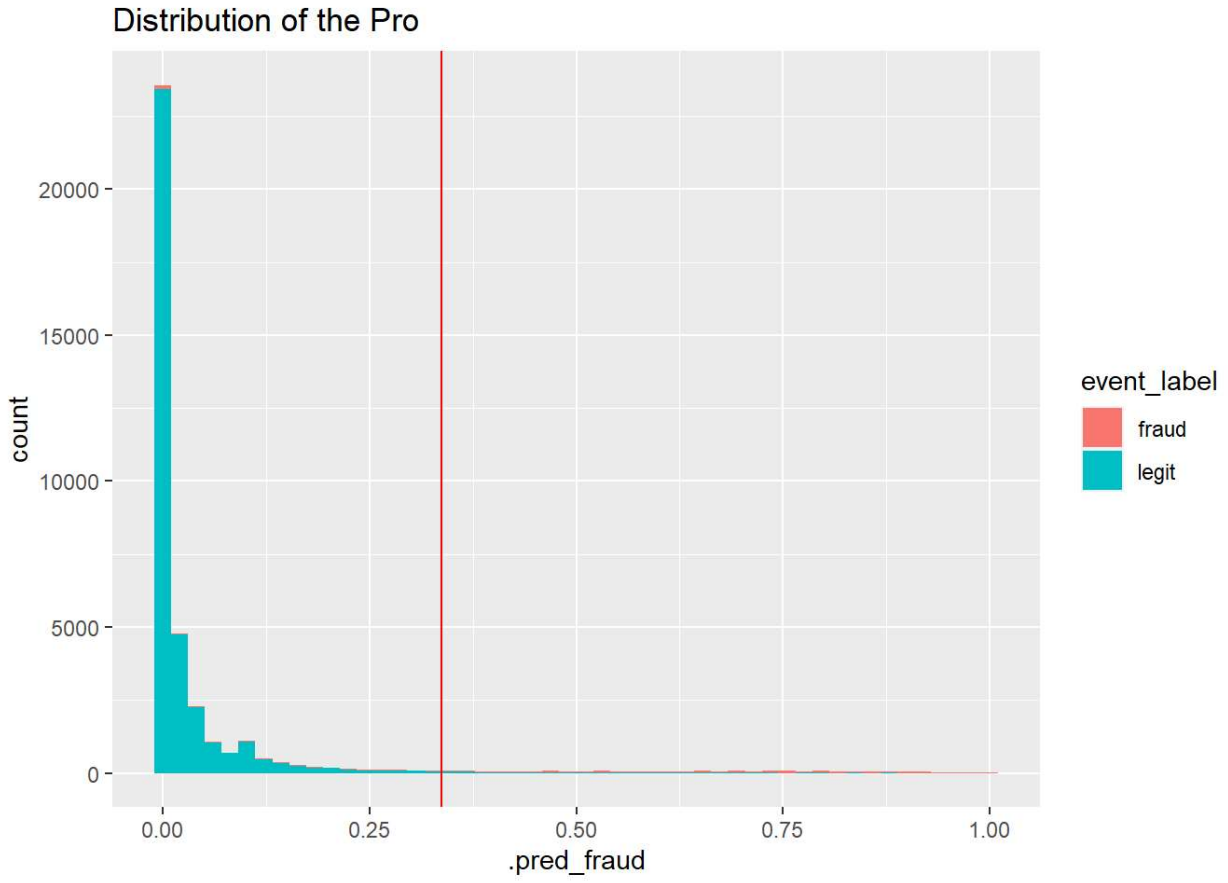


```
scored_train_2 %>%
  mutate(model = "train") %>%
  bind_rows(scored_test_2 %>%
              mutate(model="test")) %>%
  group_by(model) %>%
  spec(event_label, .pred_class) %>%
  mutate(fpr=1- .estimate)
```

model	.metric	.estimator	.estimate	fpr
<chr>	<chr>	<chr>	<dbl>	<dbl>
test	spec	binary	0.9967043	0.003295682
train	spec	binary	0.9994318	0.000568223

2 rows

```
# histogram of probability of fraud
scored_test_2 %>%
  ggplot(aes(.pred_fraud, fill = event_label)) +
  geom_histogram(bins = 50) +
  geom_vline(xintercept = 0.337, color = "red") +
  labs(title = paste("Distribution of the Pro"))
```



```
scored_train_2 %>%
  precision(event_label, .pred_class) %>%
  mutate(part="training") %>%
  bind_rows(
  scored_test_2 %>%
  precision(event_label,.pred_class) %>%
    mutate(part="testing"))
```

.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
precision	binary	0.9876218	training
precision	binary	0.9110942	testing

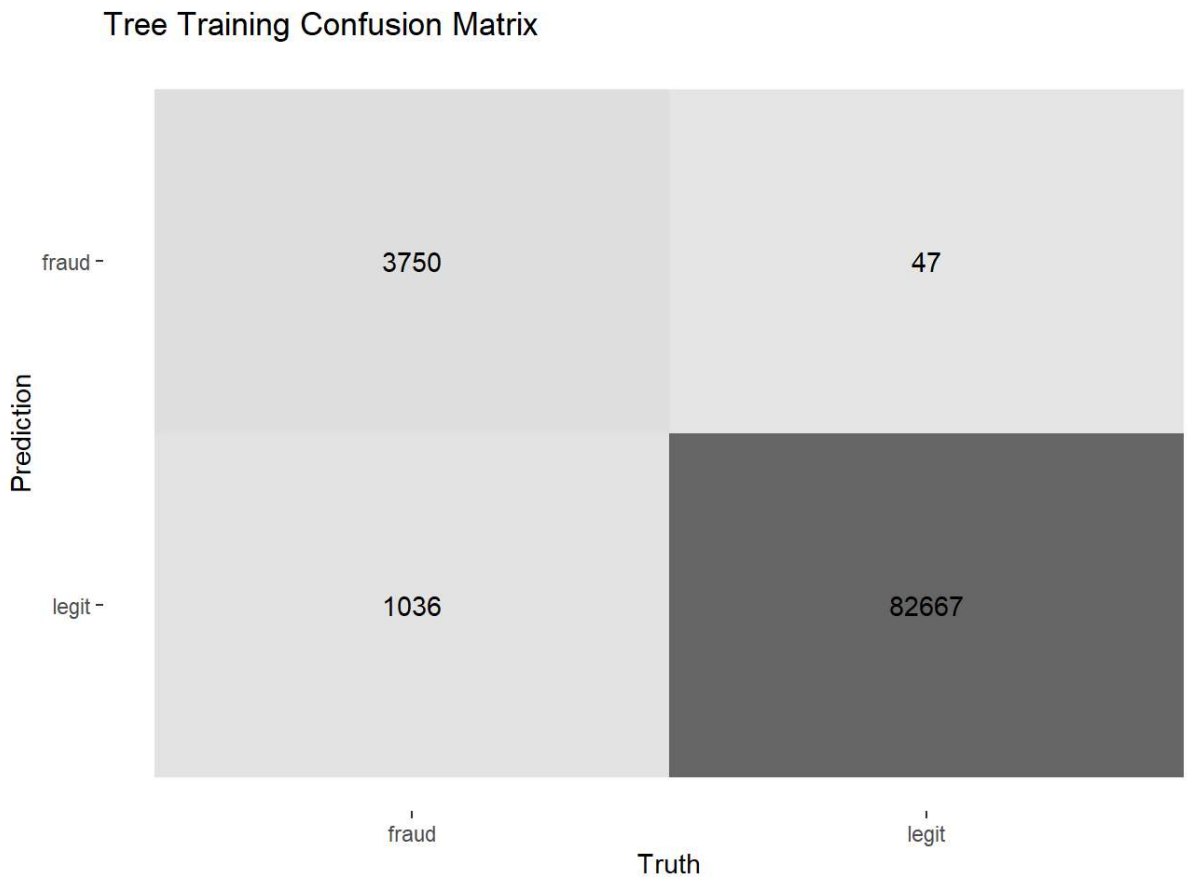
2 rows

```
scored_train_2 %>%
  recall(event_label, .pred_class) %>%
  mutate(part="training") %>%
  bind_rows(
  scored_test_2 %>%
  recall(event_label,.pred_class) %>%
    mutate(part="testing")
  )
```

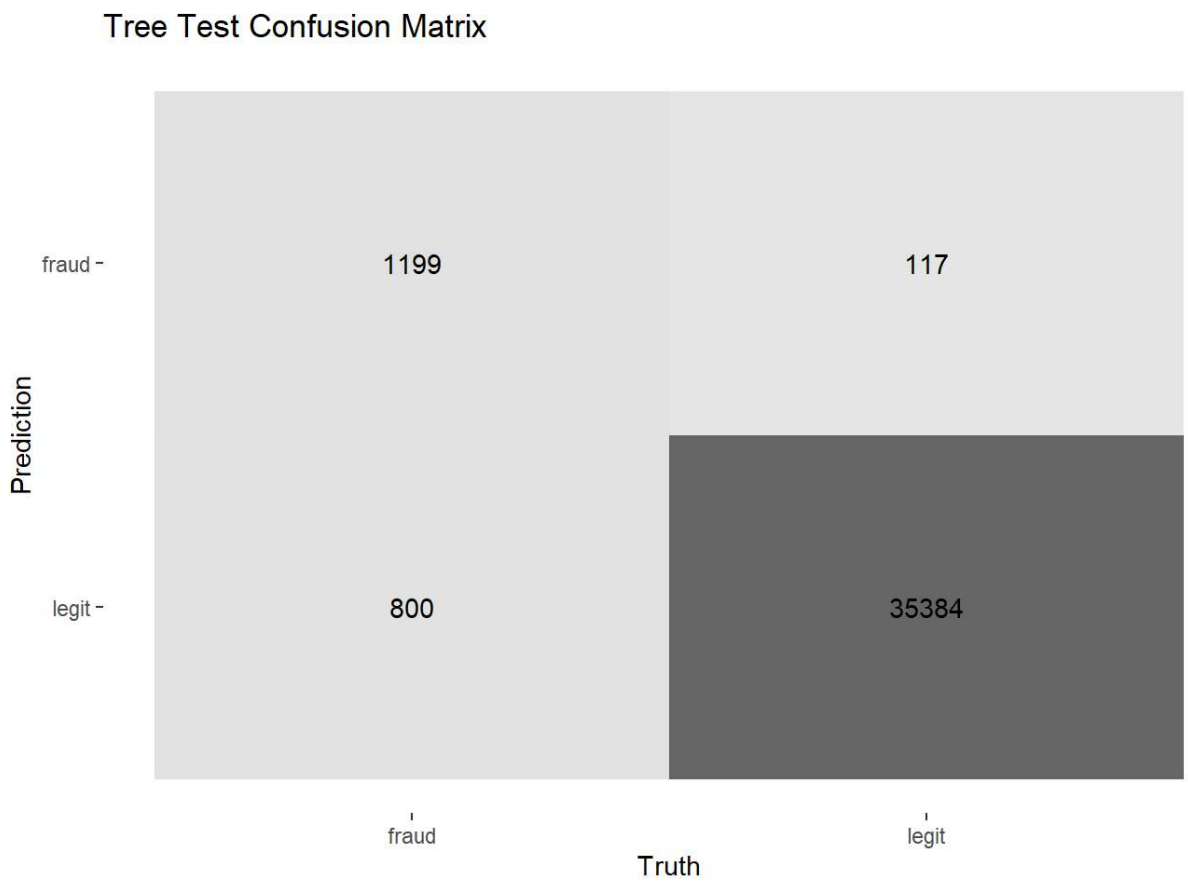
.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
recall	binary	0.7835353	training
recall	binary	0.5997999	testing

2 rows

```
scored_train_2 %>%
  conf_mat(
  truth = event_label,
  estimate = .pred_class,
  dnn = c("Prediction", "Truth")
) %>%
  autoplot(type = "heatmap") +
  labs(title="Tree Training Confusion Matrix")
```




```
scored_test_2 %>%
  conf_mat(
  truth = event_label,
  estimate = .pred_class,
  dnn = c("Prediction", "Truth")
) %>%
  autoplot(type = "heatmap") +
  labs(title="Tree Test Confusion Matrix")
```



Evaluation (rf_model_3)

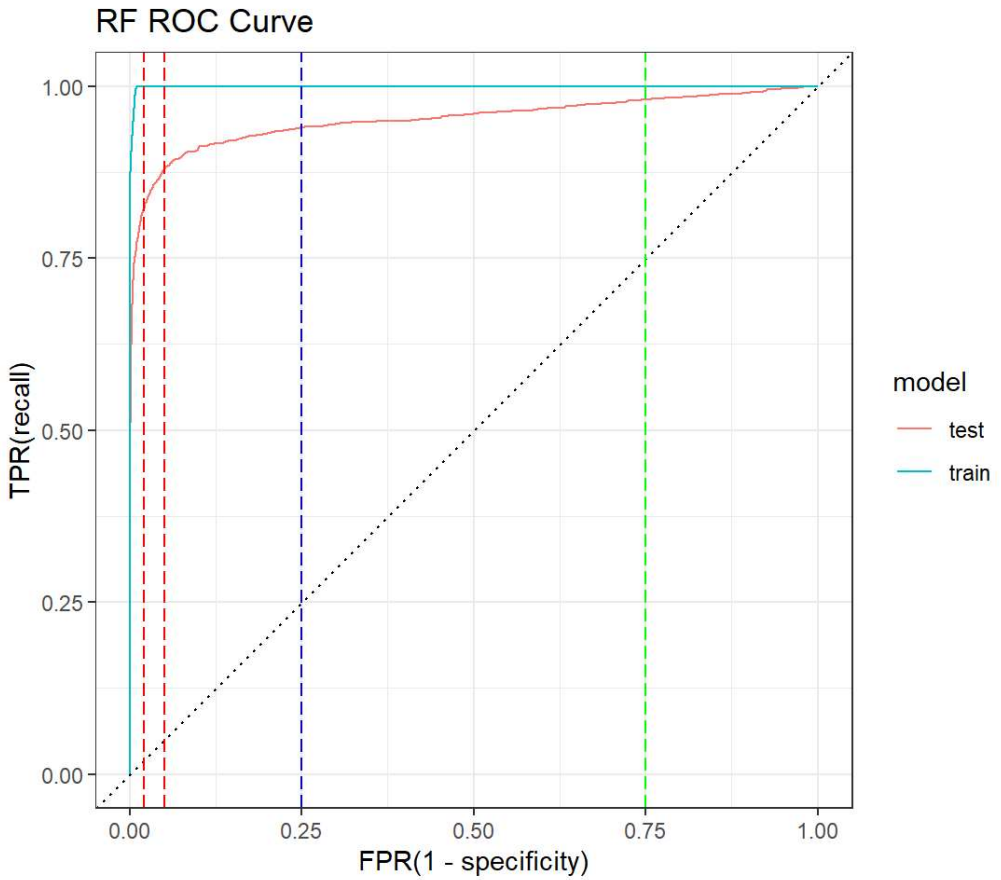
```
options(yardstick.event_first = TRUE)
  # -- score training
  predict(tree_workflow_3, train, type="prob") %>%
    bind_cols(predict(tree_workflow_3, train, type="class")) %>%
    bind_cols(., train)-> scored_train_3

  # -- score testing
  predict(tree_workflow_3, test, type="prob") %>%
      bind_cols(predict(tree_workflow_3, test, type="class")) %>%
      bind_cols(., test) -> scored_test_3
  # -- Metrics (AUC / Accuracy)
  scored_train_3 %>%
    metrics(event_label, .pred_fraud, estimate = .pred_class) %>%
    mutate(part="training") %>%
    bind_rows(scored_test_3 %>%
                metrics(event_label, .pred_fraud, estimate = .pred_class) %>%
                mutate(part="testing") ) %>%
    filter(.metric %in% c('accuracy','roc_auc')) %>%
    pivot_wider(names_from = .metric, values_from=.estimate)
```

.estimator	part	accuracy	roc_auc
<chr>	<chr>	<dbl>	<dbl>
binary	training	0.9882857	0.9993721

.estimator	part	accuracy	roc_auc
<chr>	<chr>	<dbl>	<dbl>
binary	testing	0.9773600	0.9510898

2 rows

```
# -- ROC Charts
scored_train_3 %>%
  mutate(model = "train") %>%
  bind_rows(scored_test_3 %>%
              mutate(model="test")) %>%
  group_by(model) %>%
  roc_curve(event_label, .pred_fraud) %>%
  autoplot() +
  geom_vline(xintercept = 0.0213,
             color = "red",
             linetype = "longdash") +
  geom_vline(xintercept = 0.05, # 5% FPR
             color = "red",
             linetype = "longdash") +
  geom_vline(xintercept = 0.25,    # 25% FPR
             color = "blue",
             linetype = "longdash") +
  geom_vline(xintercept = 0.75,    # 75% FPR
             color = "green",
             linetype = "longdash") +
  labs(title = "RF ROC Curve" , x = "FPR(1 - specificity)", y = "TPR(recall)")
```

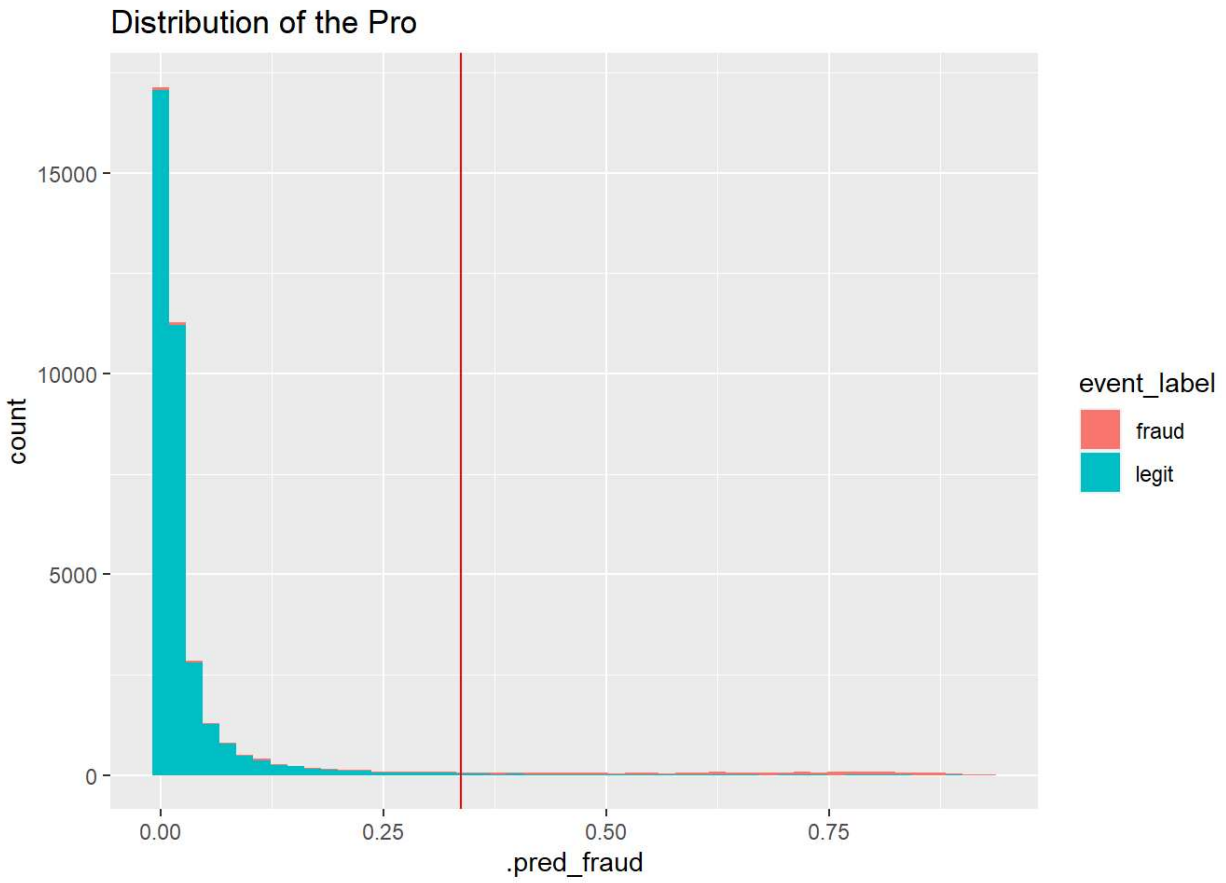


```
scored_train_3 %>%
  mutate(model = "train") %>%
  bind_rows(scored_test_3 %>%
              mutate(model="test")) %>%
  group_by(model) %>%
  spec(event_label, .pred_class) %>%
  mutate(fpr=1- .estimate)
```

model	.metric	.estimator	.estimate	fpr
<chr>	<chr>	<chr>	<dbl>	<dbl>
test	spec	binary	0.9977184	0.0022816259
train	spec	binary	0.9997945	0.0002055275

2 rows

```
# histogram of probability of fraud
scored_test_3 %>%
  ggplot(aes(.pred_fraud, fill = event_label)) +
  geom_histogram(bins = 50) +
  geom_vline(xintercept = 0.337, color = "red") +
  labs(title = paste("Distribution of the Pro"))
```



```
scored_train_3 %>%
  precision(event_label, .pred_class) %>%
  mutate(part="training") %>%
  bind_rows(
  scored_test_3 %>%
  precision(event_label,.pred_class) %>%
    mutate(part="testing"))
```

.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
precision	binary	0.9955204	training
precision	binary	0.9382622	testing

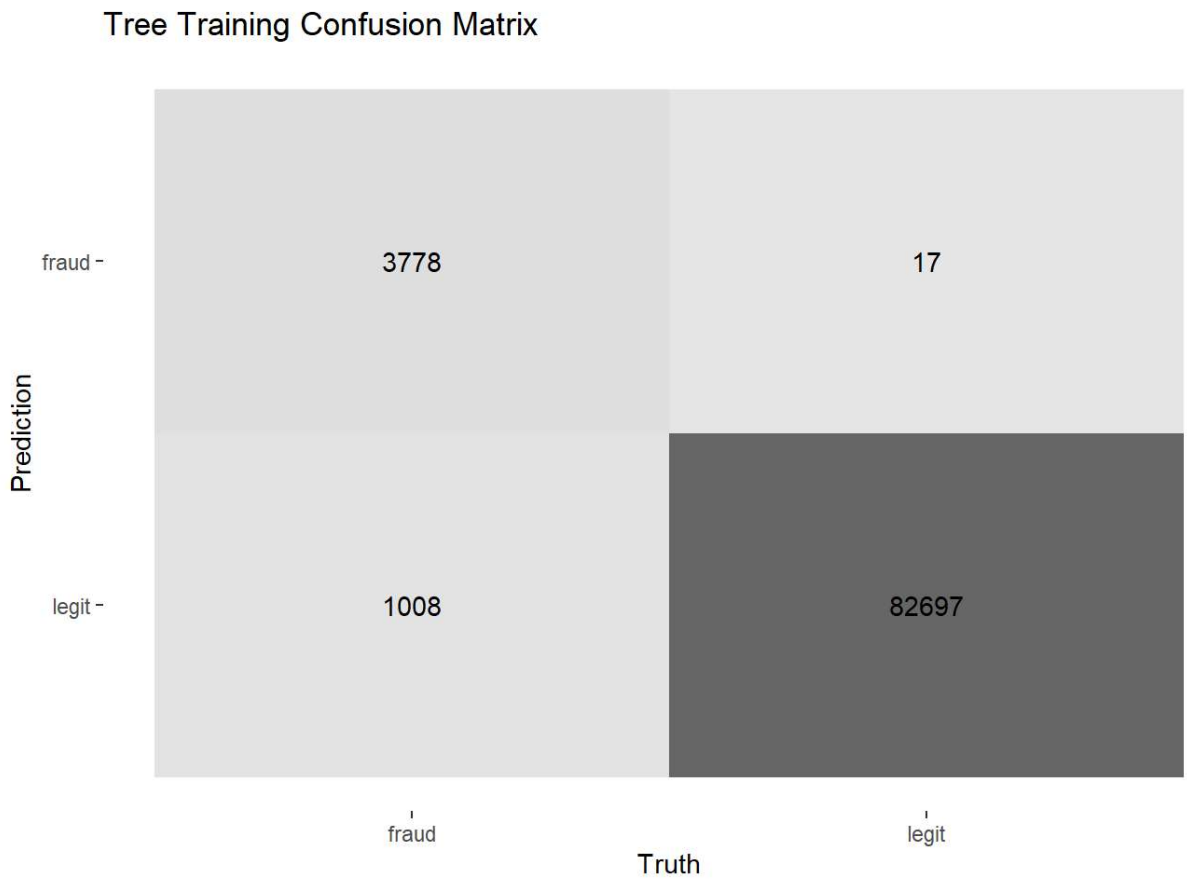
2 rows

```
scored_train_3 %>%
  recall(event_label, .pred_class) %>%
  mutate(part="training") %>%
  bind_rows(
  scored_test_3 %>%
  recall(event_label,.pred_class) %>%
    mutate(part="testing")
  )
```

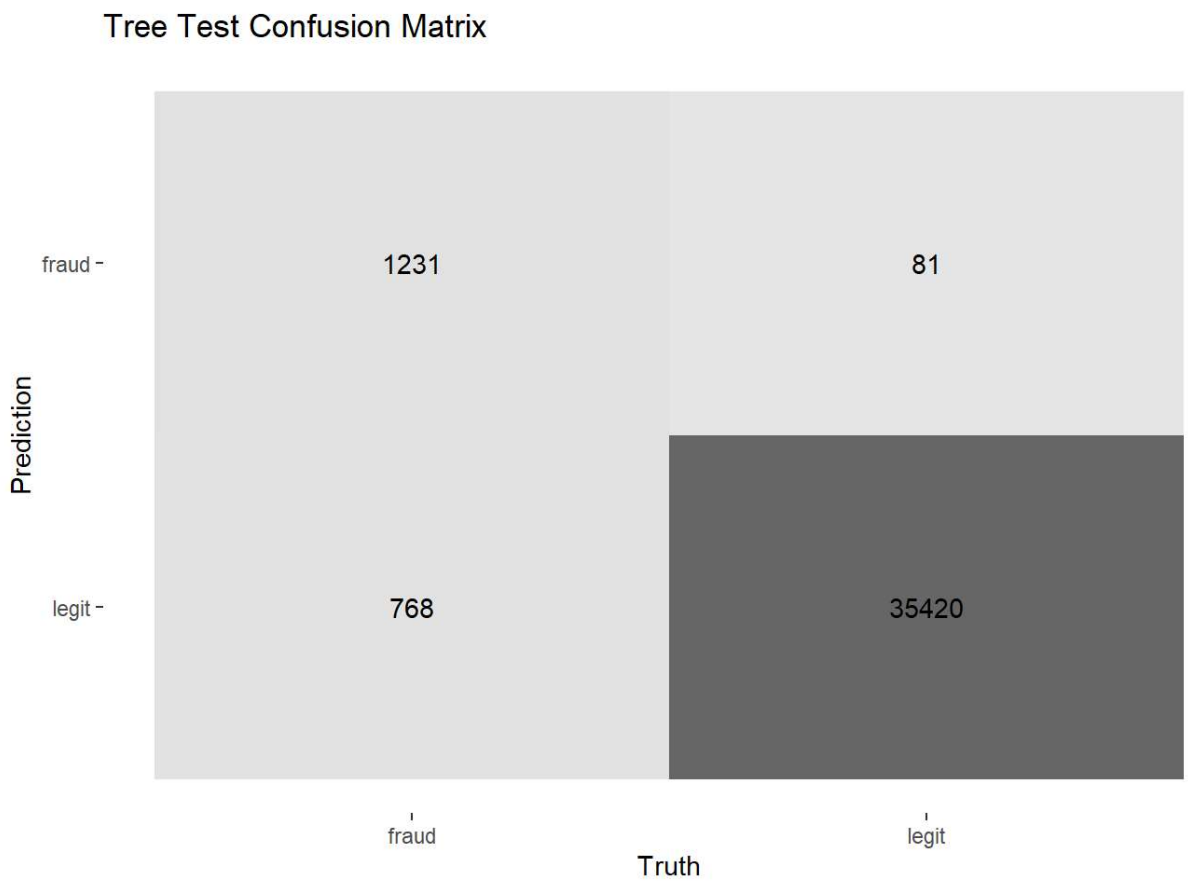
.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
recall	binary	0.7893857	training
recall	binary	0.6158079	testing

2 rows

```
scored_train_3 %>%
  conf_mat(
  truth = event_label,
  estimate = .pred_class,
  dnn = c("Prediction", "Truth")
) %>%
  autoplot(type = "heatmap") +
  labs(title="Tree Training Confusion Matrix")
```




```
scored_test_3 %>%
  conf_mat(
  truth = event_label,
  estimate = .pred_class,
  dnn = c("Prediction", "Truth")
) %>%
  autoplot(type = "heatmap") +
  labs(title="Tree Test Confusion Matrix")
```




```
scored_test_3 %>%
  roc_curve(event_label, .pred_fraud)  %>%
  mutate(
    fpr = round((1 - specificity), 2),
    tpr = round(sensitivity, 3),
    score_threshold =  round(.threshold, 3)
  ) %>%
  group_by(fpr) %>%
  summarise(threshold = round(mean(score_threshold),3),
            tpr = mean(tpr)) %>%
  filter(fpr <= 0.1)
```

fpr <dbl>	threshold <dbl>	tpr <dbl>
0.00	Inf	0.3819153
0.01	0.317	0.7628914
0.02	0.214	0.8187506
0.03	0.162	0.8455900
0.04	0.130	0.8626841
0.05	0.109	0.8781312
0.06	0.093	0.8882754
0.07	0.082	0.8943066

fpr	threshold	tpr
<dbl>	<dbl>	<dbl>
0.08	0.073	0.9012629
0.09	0.065	0.9044258

1-10 of 11 rows Previous 1 2 Next

```
scored_test_3 %>%
  mutate(fpr_1_pct = as.factor(if_else(.pred_fraud >= 0.544 ,"fraud","legit"))) %>%
  precision(event_label, fpr_1_pct)
```

.metric	.estimator	.estimate
<chr>	<chr>	<dbl>
precision	binary	0.9513014

1 row

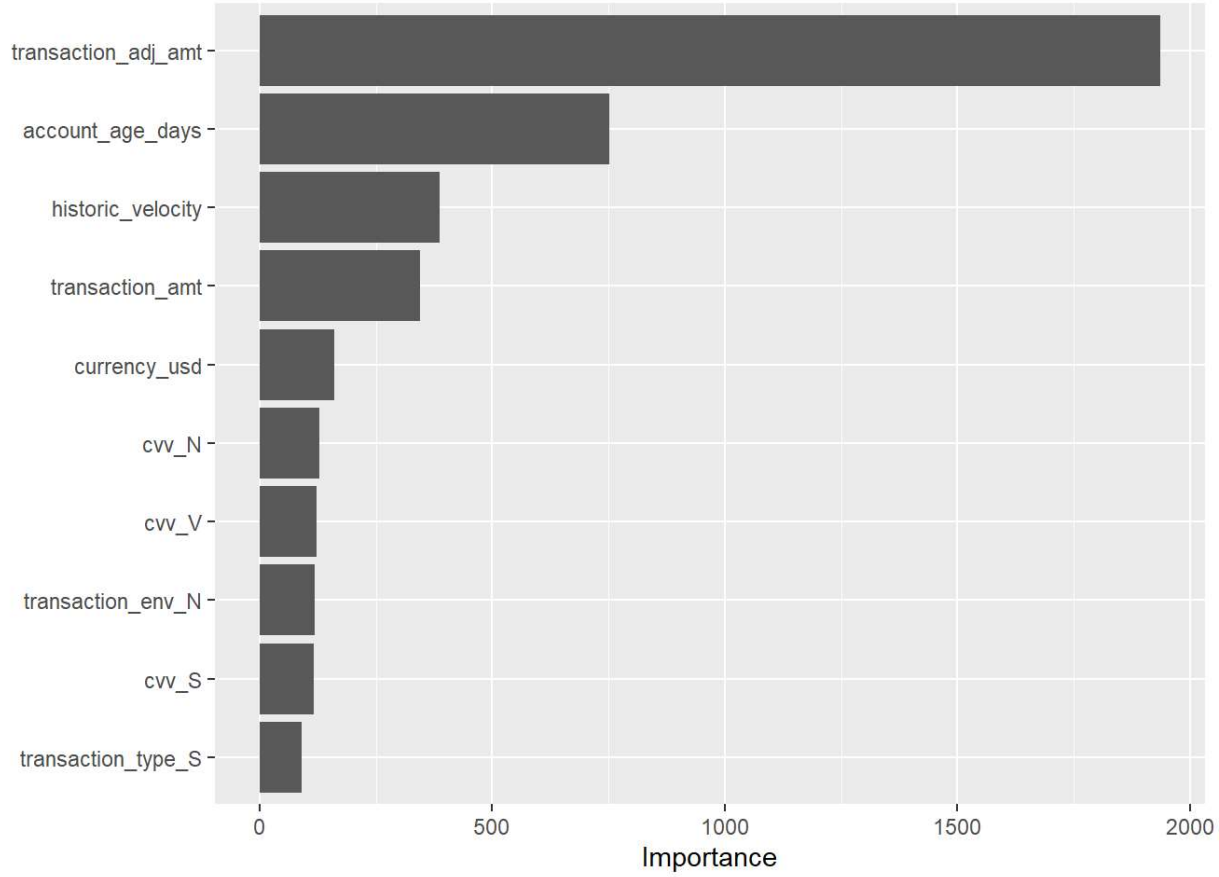
```
scored_test_3 %>%
  mutate(fpr_1_pct = as.factor(if_else(.pred_fraud >= 0.544 ,"fraud","legit"))) %>%
  recall(event_label, fpr_1_pct)
```

.metric	.estimator	.estimate
<chr>	<chr>	<dbl>
recall	binary	0.5667834

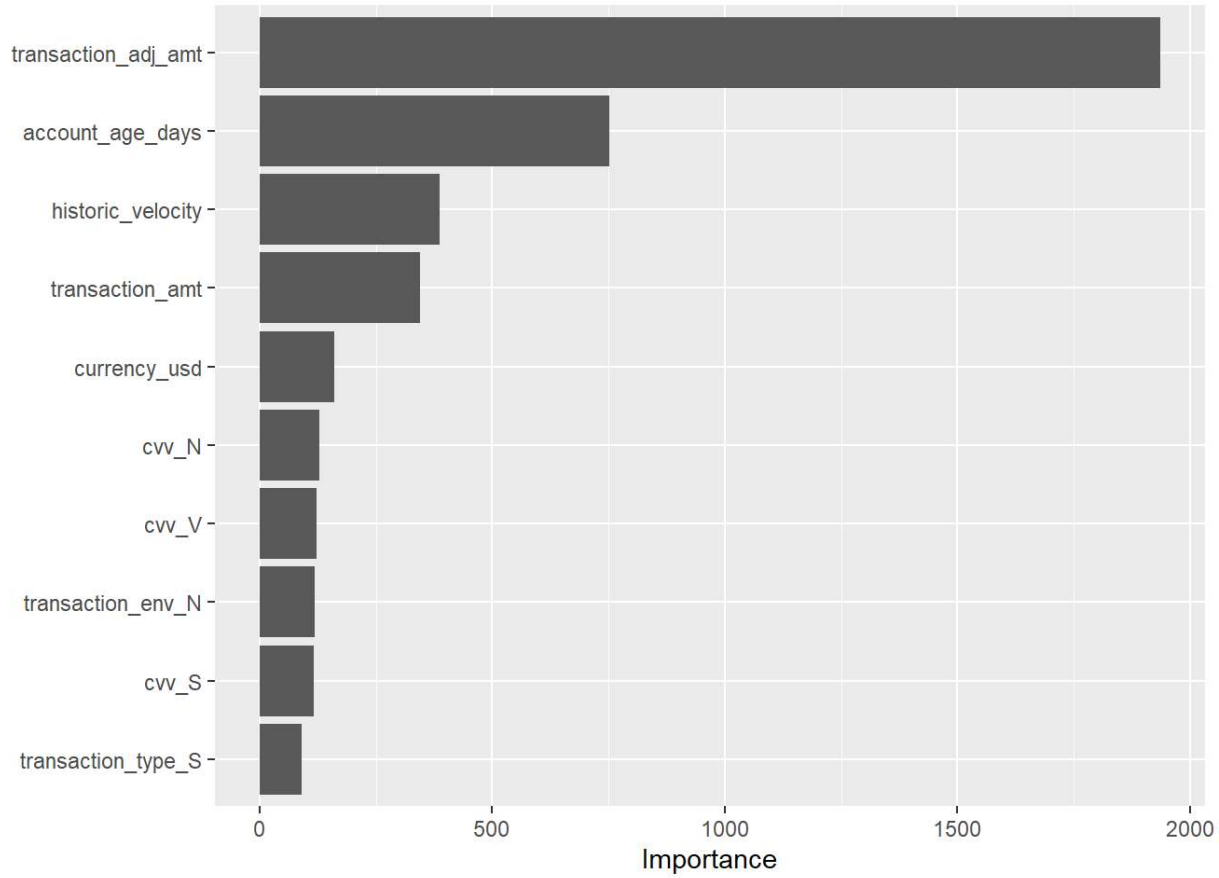
1 row

Variable Importance

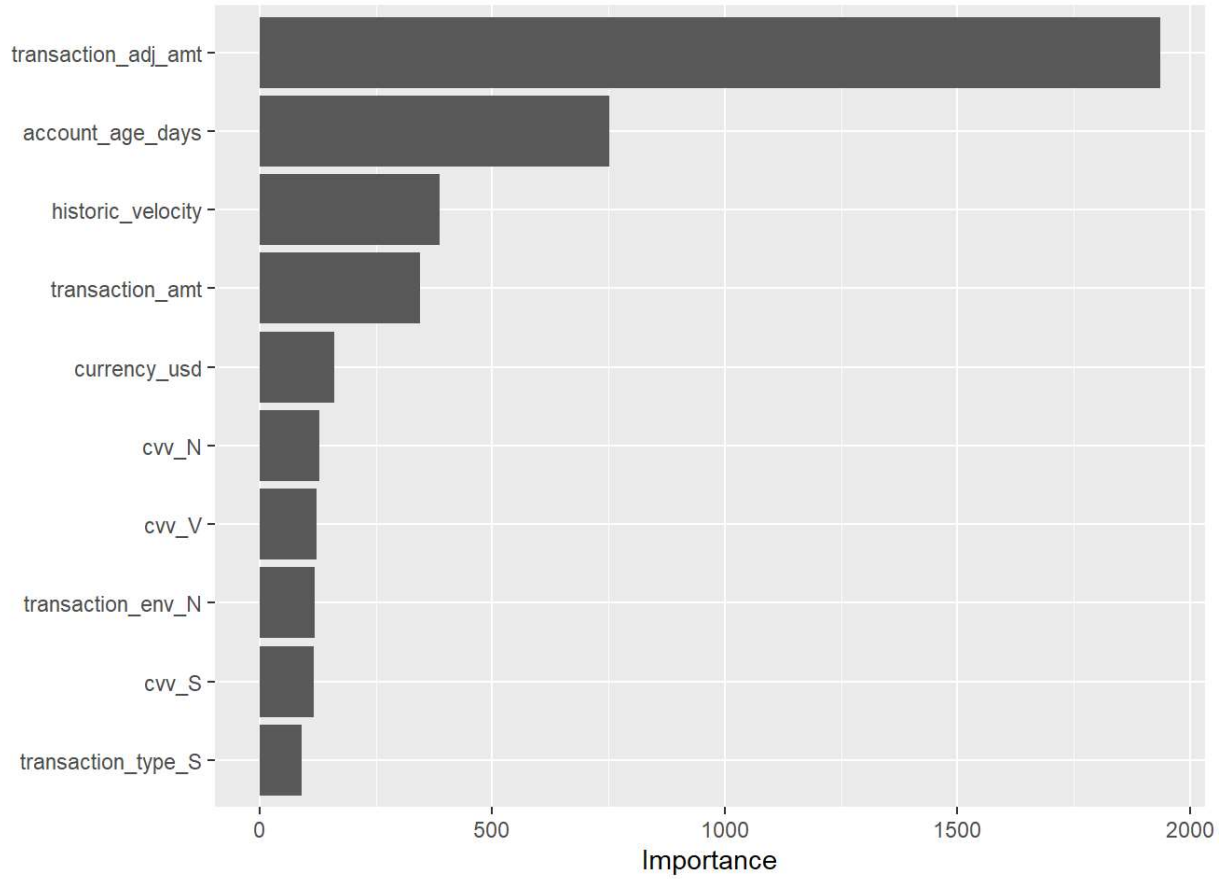
```
tree_workflow_1 %>%
  extract_fit_parsnip() %>%
  vip()
```




```
tree_workflow_1 %>%
  extract_fit_parsnip() %>%
  vip()
```



```
tree_workflow_1 %>%
  extract_fit_parsnip() %>%
  vip()
```




```
newdata <- fraud %>%
  select(account_age_days, transaction_amt, transaction_adj_amt, historic_velocity, currency, cvv, signature_image, t
         ransaction_type, transaction_env, billing_state, email_domain, event_label, billing_postal)


# -- performs our train / test split
split <- initial_split(newdata, prop = 0.7)


# -- extract the training data form our banana split
train <- training(split)
# -- extract the test data
test <- testing(split)
```

Create logistic regression for email_domain

```
email_recipe <- recipe(event_label ~ email_domain,data = train) %>%
  #step_impute_median(all_numeric_predictors()) %>% # replace numeric missing values
  step_novel(all_nominal_predictors()) %>%         # handle new levels
  themis::step_downsample(event_label, under_ratio = 3) %>%
  step_unknown(all_nominal_predictors()) %>%       # replace category missing values
  step_other(all_nominal_predictors(),threshold = 10) %>%  # pool rarely occuring levels
  step_dummy(all_nominal_predictors(), one_hot = TRUE) # onehot encode

bake(email_recipe %>% prep(), train %>% sample_n(1000))
```

event_label	email_domain_allison.castaneda.info	email_domain_anderson.bradley.biz
<fct>	<int>	<int>
legit	0	0
legit	0	0
fraud	0	0
legit	0	0
legit	0	0
legit	0	0
fraud	0	0
legit	0	0
legit	0	0
legit	0	0

1-10 of 1,000 rows | 1-3 of 138 columns Previous **1** 2 3 4 5 6 ... 100 Next

```
email_model <- logistic_reg() %>%
   set_mode("classification") %>%
   set_engine("glm")

email_workflow <- workflow() %>%
  add_recipe(email_recipe) %>%
  add_model(email_model) %>%
  fit(train)

tidy(email_workflow) %>%
  mutate_if(is.numeric,round,3) %>%
  filter(p.value < 0.05)
```

term	estimate	std.error	statistic	p.value
<chr>	<dbl>	<dbl>	<dbl>	<dbl>
(Intercept)	1.125	0.018	64.205	0.000
email_domain_davis.white.com	-1.972	0.690	-2.857	0.004
email_domain_frazier.woods.info	-1.530	0.646	-2.370	0.018
email_domain_hernandez.owens.com	-1.307	0.606	-2.158	0.031
email_domain_huffman.org	-1.307	0.606	-2.158	0.031
email_domain_lewis.boyd.net	-1.530	0.646	-2.370	0.018
email_domain_lloyd.lane.com	-1.530	0.646	-2.370	0.018
email_domain_munoz.miranda.com	-1.307	0.606	-2.158	0.031
email_domain_ochoa.jones.net	-1.307	0.606	-2.158	0.031
email_domain_pham.stone.com	-1.530	0.646	-2.370	0.018

1-10 of 12 rows Previous **1** 2 Next

```
  options(yardstick.event_first = TRUE)

# score training
predict(email_workflow, train, type = "prob") %>%
  bind_cols(predict(email_workflow, train, type = "class")) %>%
  mutate(part = "train") %>%
  bind_cols(., train) -> scored_train_email

# -- score testing
predict(email_workflow, test, type = "prob") %>%
  bind_cols(predict(email_workflow,  test, type = "class")) %>%
  mutate(part = "testing") %>%
  bind_cols(., test) -> scored_test_email

## Metrics (AUC / Accuracy / Log Loss)
bind_rows (scored_train_email, scored_test_email)  %>%
  group_by(part) %>%
  metrics(event_label, .pred_fraud, estimate = .pred_class) %>%
  filter(.metric %in% c('accuracy', 'roc_auc')) %>%
  pivot_wider(names_from = .metric, values_from = .estimate)
```

part	.estimator	accuracy	roc_auc
<chr>	<chr>	<dbl>	<dbl>
testing	binary	0.9432267	0.5046562
train	binary	0.9434629	0.5245122

2 rows

```
# precision @0.5
bind_rows(scored_train_email, scored_test_email) %>%
  group_by(part) %>%
  precision(event_label, .pred_class)
```

part	.metric	.estimator	.estimate
<chr>	<chr>	<chr>	<dbl>
testing	precision	binary	0.06299213
train	precision	binary	0.21337580

2 rows

```
# recall @0.5
bind_rows(scored_train_email, scored_test_email) %>%
  group_by(part) %>%
  recall(event_label, .pred_class)
```

part	.metric	.estimator	.estimate
<chr>	<chr>	<chr>	<dbl>
testing	recall	binary	0.003964321
train	recall	binary	0.014054961

2 rows

Create logistic regression for billing_postal

```
billing_recipe <- recipe(event_label ~ billing_postal,data = train) %>%
  #step_impute_median(all_numeric_predictors()) %>% # replace numeric missing values
  step_novel(all_nominal_predictors()) %>%         # handle new levels
  themis::step_downsample(event_label, under_ratio = 3) %>%
  step_unknown(all_nominal_predictors()) %>%       # replace category missing values
  step_other(all_nominal_predictors(),threshold = 10) %>%  # pool rarely occuring levels
  step_dummy(all_nominal_predictors(), one_hot = TRUE) # onehot encode

bake(billing_recipe %>% prep(), train %>% sample_n(1000))
```

billing_postal	event_label
<dbl>	<fct>
54010	legit
68502	legit
94454	legit
18050	legit

billing_postal	event_label
<dbl>	<fct>
80182	legit
19680	legit
65733	legit
65207	legit
45708	legit
42569	legit

1-10 of 1,000 rows Previous **1** 2 3 4 5 6 ... 100 Next

```
billing_model <- logistic_reg() %>%
   set_mode("classification") %>%
   set_engine("glm")

billing_workflow <- workflow() %>%
  add_recipe(billing_recipe) %>%
  add_model(billing_model) %>%
  fit(train)

tidy(billing_workflow) %>%
  mutate_if(is.numeric,round,3) %>%
  filter(p.value < 0.05)
```

term	estimate	std.error	statistic	p.value
<chr>	<dbl>	<dbl>	<dbl>	<dbl>
(Intercept)	1.054	0.034	31.282	0

1 row

```
  options(yardstick.event_first = TRUE)

# score training
predict(billing_workflow, train, type = "prob") %>%
  bind_cols(predict(billing_workflow, train, type = "class")) %>%
  mutate(part = "train") %>%
  bind_cols(., train) -> scored_train_billing

# -- score testing
predict(billing_workflow, test, type = "prob") %>%
  bind_cols(predict(billing_workflow,  test, type = "class")) %>%
  mutate(part = "testing") %>%
  bind_cols(., test) -> scored_test_billing

## Metrics (AUC / Accuracy / Log Loss)
bind_rows (scored_train_billing, scored_test_billing)  %>%
  group_by(part) %>%
  metrics(event_label, .pred_fraud, estimate = .pred_class) %>%
  filter(.metric %in% c('accuracy', 'roc_auc')) %>%
  pivot_wider(names_from = .metric, values_from = .estimate)
```

part	.estimator	accuracy	roc_auc
<chr>	<chr>	<dbl>	<dbl>
testing	binary	0.9461602	0.4949085
train	binary	0.9455506	0.5063284

2 rows

```
# precision @0.5
bind_rows(scored_train_email, scored_test_billing) %>%
  group_by(part) %>%
  precision(event_label, .pred_class)
```

part	.metric	.estimator	.estimate
<chr>	<chr>	<chr>	<dbl>
testing	precision	binary	*NA*
train	precision	binary	0.2133758

2 rows

```
# recall @0.5
bind_rows(scored_train_email, scored_test_billing) %>%
  group_by(part) %>%
  recall(event_label, .pred_class)
```

part	.metric	.estimator	.estimate
<chr>	<chr>	<chr>	<dbl>
testing	recall	binary	0.00000000
train	recall	binary	0.01405496

2 rows

10. Kaggle

```
# -- score testing
predict(tree_workflow_3, fraud_kaggle, type="prob") %>%
       bind_cols(., fraud_kaggle) %>%
  select(event_id, event_label=.pred_fraud) -> kaggle_prediction

kaggle_prediction %>%
  write_csv("challenge_2_kaggle.csv")
```