Final Loan Default

Eagle Xuhui Ying

12/09/2022

Load Libraries

```
options(warn = -1)
options(scipen = 999) # turns off scientific notation
library(tidyverse)
library(tidymodels)
library(dplyr)
library(janitor)
library(solitude) # Isolation Forest
library(ggpubr) # Isolation Forest
library(skimr)
library(modelr)
library(GGally)
library(kableExtra) # make nice looking results when we knit
library(fastshap)   # shapley values for variable importance
library(MASS)
library(tree)
library(ggplot2)
library(corrplot)
library(factoextra)
library(rpart.plot) # plotting decision trees
library(lubridate)
library(vip)
library(NeuralNetTools) # visualization of neural networks
library(reshape2)
library(PerformanceAnalytics)
library(DALEX)    # new
library(DALEXtra) # new
library(caret)
```

Import Data

```
loan <- read_csv("loan_train.csv") %>% clean_names() %>%
  mutate(issue_d_year = year(Sys.Date()) - year(my(issue_d))) %>%
  mutate(earliest_cr_line_year = year(Sys.Date()) - year(my(earliest_cr_line))) %>%
  mutate(last_pymnt_d_year = year(Sys.Date()) - year(my(last_pymnt_d))) %>%
  mutate(last_credit_pull_d_year = year(Sys.Date()) - year(my(last_credit_pull_d))) %>%
  dplyr::select(-issue_d, -earliest_cr_line, -last_pymnt_d, -last_credit_pull_d)
head(loan)
```

id	member...	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	gr...	sub_grade
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<chr>	<dbl>	<chr>	<chr>
1077501	1296599	5000	5000	4975	36 months	10.65%	162.87	B	B2
1077430	1314167	2500	2500	2500	60 months	15.27%	59.83	C	C4
1076863	1277178	10000	10000	10000	36 months	13.49%	339.31	C	C1
1069639	1304742	7000	7000	7000	60 months	15.96%	170.08	C	C5
1072053	1288686	3000	3000	3000	36 months	18.64%	109.43	E	E1
1071795	1306957	5600	5600	5600	60 months	21.28%	152.39	F	F2

6 rows | 1-10 of 52 columns

```
nrow(loan)
```

```
## [1] 29777
```

```
skim(loan)
```

Data summary

Name	loan
Number of rows	29777
Number of columns	52
_____	
Column type frequency:	
character	19
numeric	33
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
term	3	1.00	9	9	0	2	0
int_rate	3	1.00	5	6	0	390	0

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
grade	3	1.00	1	1	0	7	0
sub_grade	3	1.00	2	2	0	35	0
emp_title	1817	0.94	2	75	0	22143	0
emp_length	3	1.00	3	9	0	12	0
home_ownership	3	1.00	3	8	0	5	0
verification_status	3	1.00	8	15	0	3	0
loan_status	0	1.00	7	7	0	2	0
pymnt_plan	3	1.00	1	1	0	2	0
url	3	1.00	62	64	0	29774	0
desc	9432	0.68	13	943	0	20310	0
purpose	3	1.00	3	18	0	14	0
title	13	1.00	2	80	0	15200	0
zip_code	3	1.00	5	5	0	819	0
addr_state	3	1.00	2	2	0	50	0
revol_util	67	1.00	2	6	0	1094	0
next_pymnt_d	27425	0.08	8	8	0	96	0
application_type	3	1.00	10	10	0	1	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
id	3	1.00	663006.18	219881.52	54734.00	496344.25	642763.00	825647.50	1077501.00	▂▅▇
member_id	3	1.00	823568.15	280420.33	70473.00	635380.00	822247.50	1033656.25	1314167.00	▂▅▇
loan_amnt	3	1.00	11109.43	7404.65	500.00	5225.00	9800.00	15000.00	35000.00	▇▃▂▁
funded_amnt	3	1.00	10843.64	7147.05	500.00	5100.00	9600.00	15000.00	35000.00	▇▃▂▁
funded_amnt_inv	3	1.00	10149.66	7130.86	0.00	4950.00	8500.00	14000.00	35000.00	▇▃▂▁
installment	3	1.00	323.81	209.77	15.67	165.84	278.94	429.86	1305.19	▇▃▁
annual_inc	4	1.00	69201.23	66566.42	2000.00	40000.00	59000.00	82500.00	6000000.00	▇▁
dti	3	1.00	13.38	6.74	0.00	8.19	13.49	18.70	29.99	▆▇▆
delinq_2yrs	23	1.00	0.16	0.52	0.00	0.00	0.00	0.00	13.00	▇▁
fico_range_low	3	1.00	713.05	36.31	610.00	685.00	710.00	740.00	825.00	▃▇▃▁
fico_range_high	3	1.00	717.05	36.31	614.00	689.00	714.00	744.00	829.00	▃▇▃▁
inq_last_6mths	23	1.00	1.08	1.54	0.00	0.00	1.00	2.00	33.00	▇▁
mths_since_last_delinq	18907	0.37	34.72	22.32	0.00	17.00	32.00	51.00	120.00	▇▇▃▁
mths_since_last_record	27208	0.09	59.23	47.22	0.00	0.00	85.00	102.00	129.00	▇▁▂▇
open_acc	23	1.00	9.34	4.51	1.00	6.00	9.00	12.00	47.00	▇▃▁
pub_rec	23	1.00	0.06	0.25	0.00	0.00	0.00	0.00	5.00	▇▁
revol_bal	3	1.00	14310.00	22696.55	0.00	3644.00	8783.50	17187.00	1207359.00	▇▁
total_acc	23	1.00	22.08	11.59	1.00	13.00	20.00	29.00	81.00	▇▆▂▁
out_prncp	3	1.00	11.80	123.53	0.00	0.00	0.00	0.00	3126.61	▇▁
out_prncp_inv	3	1.00	11.76	123.23	0.00	0.00	0.00	0.00	3123.44	▇▁
total_rec_late_fee	3	1.00	1.50	7.72	0.00	0.00	0.00	0.00	180.20	▇▁
last_pymnt_amnt	3	1.00	2615.41	4373.70	0.00	211.02	531.98	3171.29	36115.20	▇▁
collections_12_mths_ex_med	104	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	▁▇▁
policy_code	3	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00	▁▇▁
acc_now_delinq	23	1.00	0.00	0.01	0.00	0.00	0.00	0.00	1.00	▇▁
chargeoff_within_12_mths	104	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	▁▇▁
delinq_amnt	23	1.00	0.20	35.09	0.00	0.00	0.00	0.00	6053.00	▇▁
pub_rec_bankruptcies	966	0.97	0.05	0.21	0.00	0.00	0.00	0.00	2.00	▇▁
tax_liens	79	1.00	0.00	0.01	0.00	0.00	0.00	0.00	1.00	▇▁

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
issue_d_year	3	1.00	11.78	0.97	11.00	11.00	11.00	12.00	15.00	▇▂▁▁▁
earliest_cr_line_year	23	1.00	25.43	6.85	14.00	21.00	24.00	29.00	76.00	▇▂▁▁▁
last_pymnt_d_year	67	1.00	9.30	1.61	6.00	8.00	9.00	10.00	15.00	▇▆▂▁▁
last_credit_pull_d_year	5	1.00	7.43	1.81	6.00	6.00	6.00	9.00	15.00	▇▂▁▁▁

```
# drop columns: desc, next_pymnt_d, mths_since_last_delinq, mths_since_last_record (>20% missing values)

loan_kaggle <- read_csv("loan_holdout.csv") %>% clean_names() %>%
  mutate(issue_d_year = year(Sys.Date()) - year(my(issue_d))) %>%
  mutate(earliest_cr_line_year = year(Sys.Date()) - year(my(earliest_cr_line))) %>%
  mutate(last_pymnt_d_year = year(Sys.Date()) - year(my(last_pymnt_d))) %>%
  mutate(last_credit_pull_d_year = year(Sys.Date()) - year(my(last_credit_pull_d))) %>%
  dplyr::select(-issue_d, -earliest_cr_line, -last_pymnt_d, -last_credit_pull_d)
head(loan_kaggle)
```

id	member...	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	gr...	sub_grade
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<chr>	<dbl>	<chr>	<chr>
1077175	1313524	2400	2400	2400	36 months	15.96%	84.33	C	C5
1075358	1311748	3000	3000	3000	60 months	12.69%	67.79	B	B5
1075269	1311441	5000	5000	5000	36 months	7.90%	156.46	A	A4
1071570	1306721	5375	5375	5350	60 months	12.69%	121.45	B	B5
1064687	1298717	9000	9000	9000	36 months	13.49%	305.38	C	C1
1065775	1299699	10000	10000	10000	36 months	15.27%	347.98	C	C4

6 rows | 1-10 of 51 columns

Explanatory Data Analysis

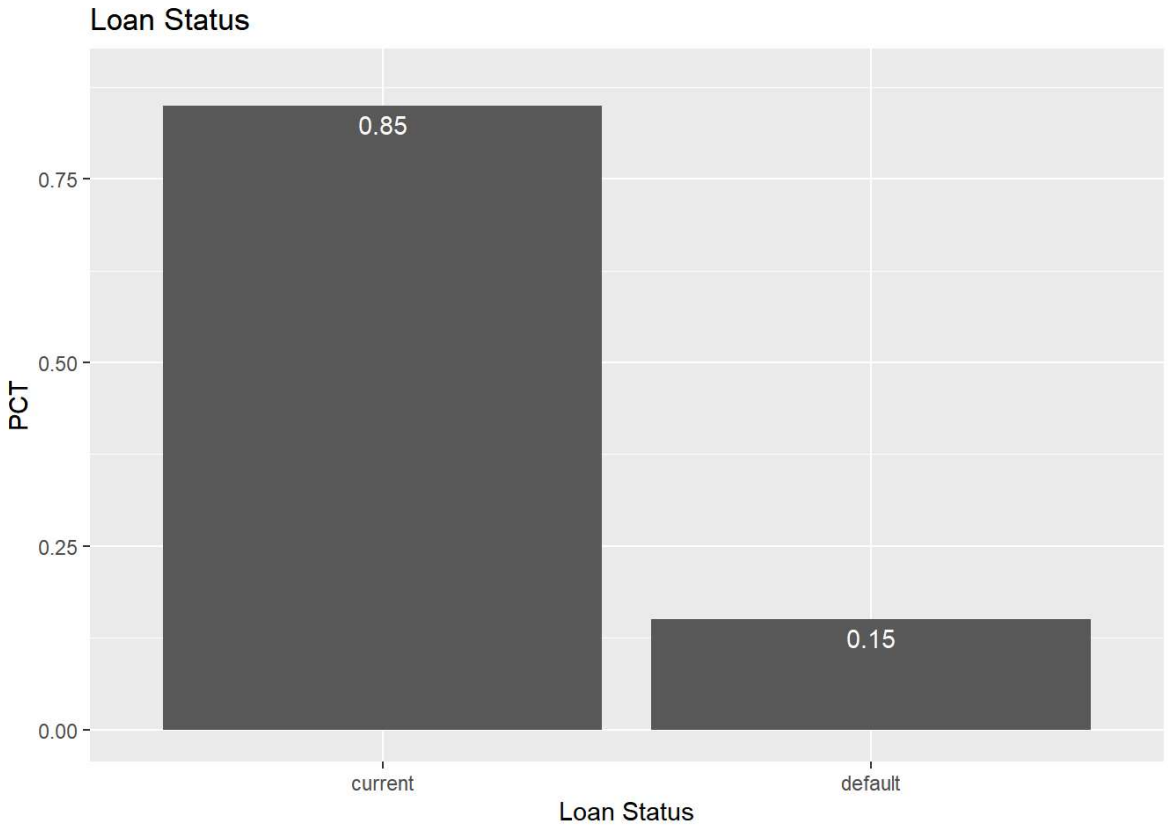
Explore Target

```
loan_summary <- loan %>%
  count(loan_status) %>%
  mutate(pct = n/sum(n))

loan_summary
```

loan_status	n	pct
<chr>	<int>	<dbl>
current	25300	0.8496491
default	4477	0.1503509

2 rows

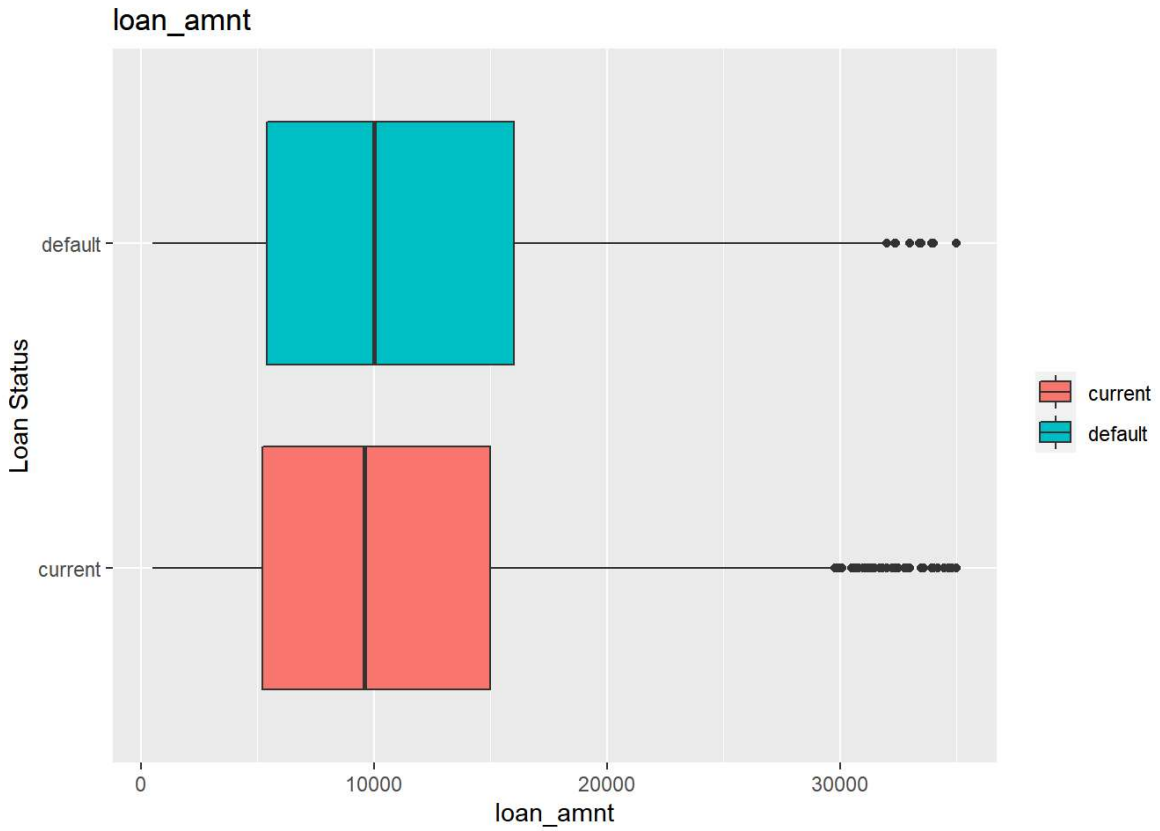
```
loan_summary %>%
  ggplot(aes(x=factor(loan_status),y=pct)) +
  geom_col()  +
  geom_text(aes(x=factor(loan_status), y=pct+0.034, label=round(pct,2)), vjust=2.75, colour="white") +
  labs(title="Loan Status", x="Loan Status", y="PCT")
```

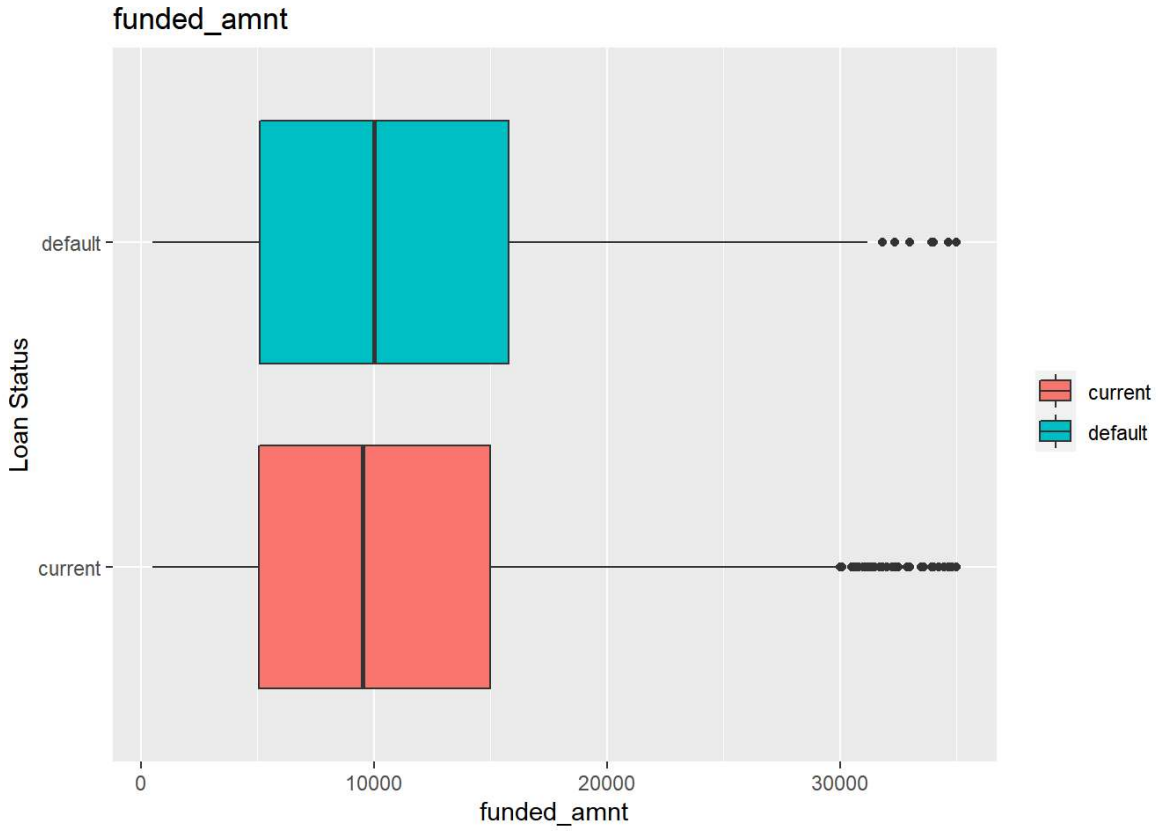


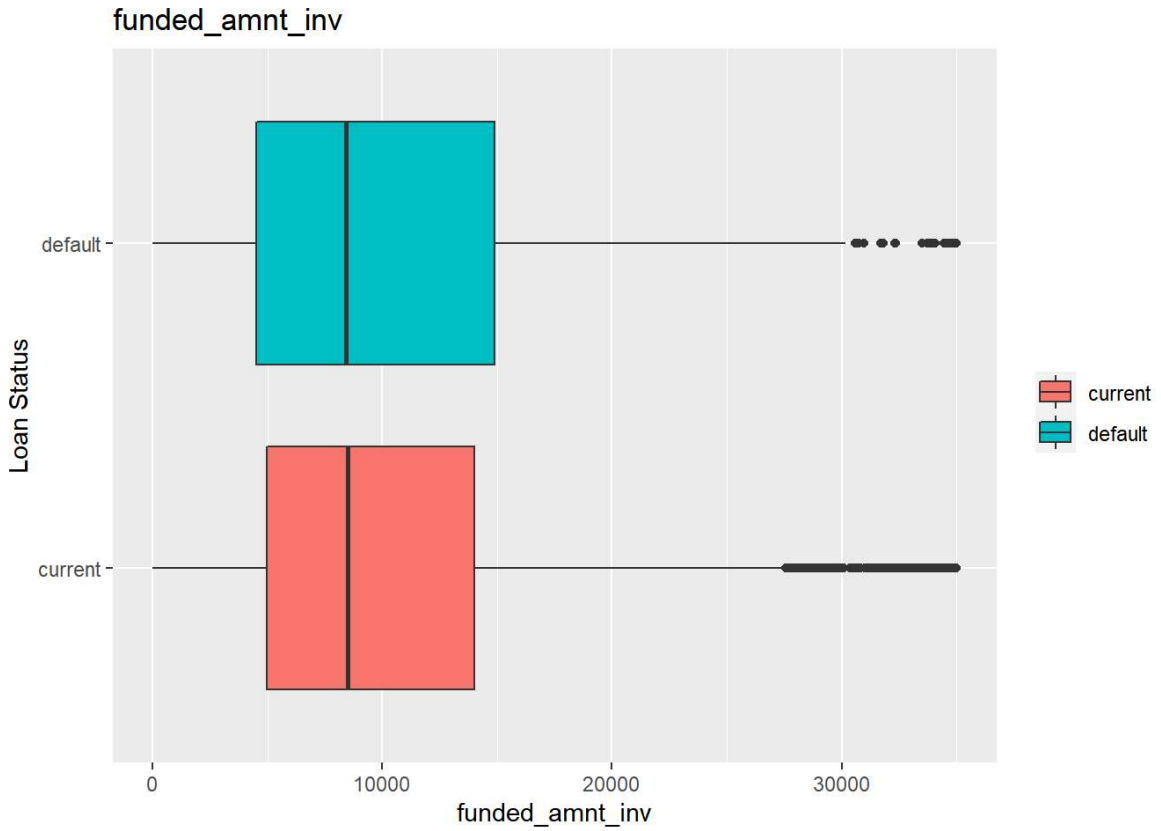
Explore Numerics

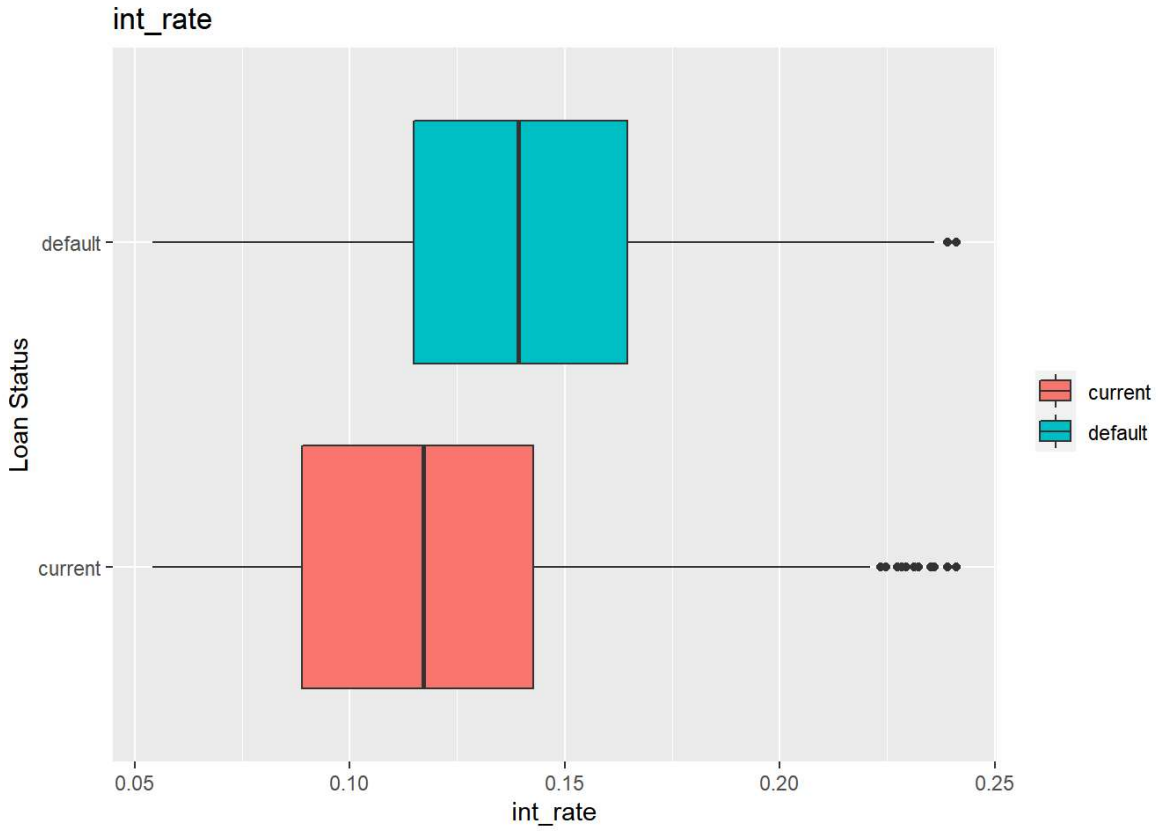
numeric variables: loan_amnt, funded_amnt, funded_amnt_inv, int_rate, installment, annual_inc, dti, delinq_2yrs, fico_range_low, fico_range_high, inq_last_6mths, open_acc, revol_bal, revol_util, total_acc, out_prncp, out_prncp_inv, total_rec_late_fee, total_rec_late_fee, last_pymnt_amnt, issue_d_year, earliest_cr_line_year, last_pymnt_d_year, last_credit_pull_d_year

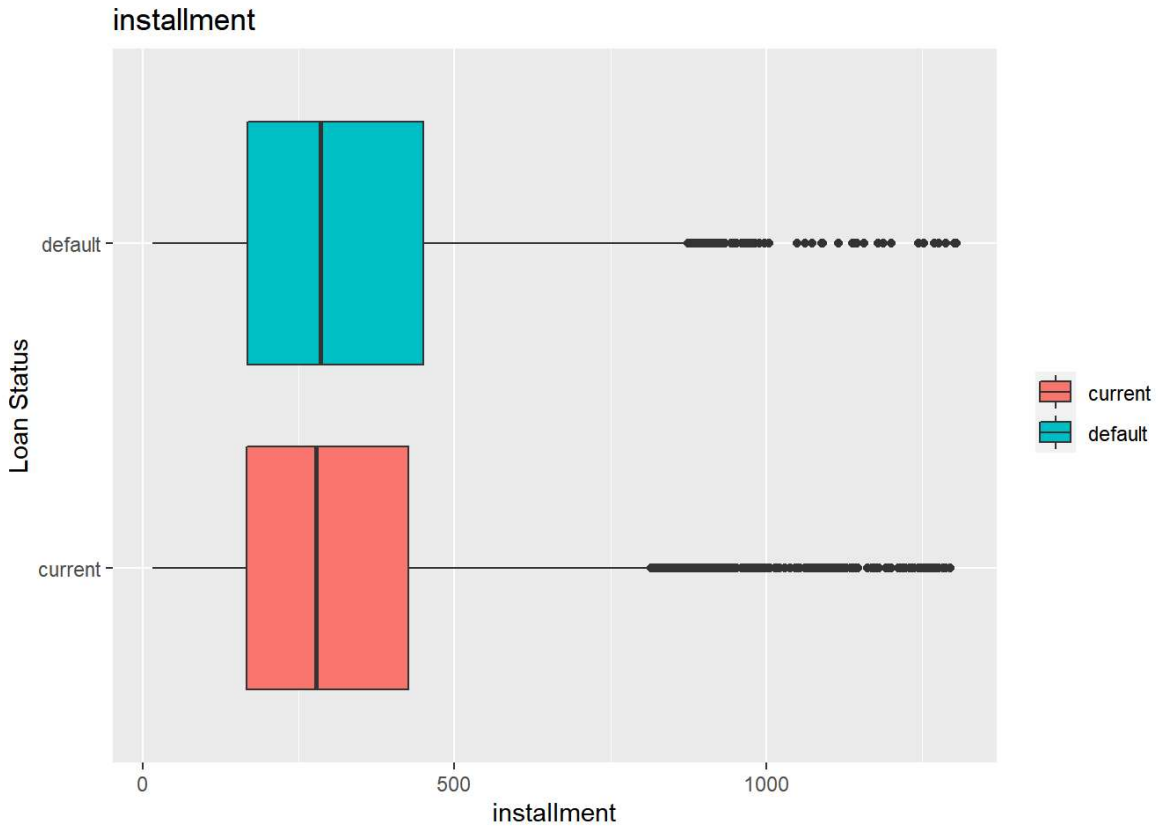
```
# convert percentage to decimal

loan_decimal <- loan
loan_decimal$int_rate <- as.numeric(lapply(loan$int_rate, function(x) as.numeric(sub("%", "", x))/100))
loan_decimal$revol_util <- as.numeric(lapply(loan$revol_util, function(x) as.numeric(sub("%", "", x))/100))

loan_kaggle_decimal <- loan_kaggle
loan_kaggle_decimal$int_rate <- as.numeric(lapply(loan_kaggle$int_rate, function(x) as.numeric(sub("%", "", x))/100))
loan_kaggle_decimal$revol_util <- as.numeric(lapply(loan_kaggle$revol_util, function(x) as.numeric(sub("%", "", x))/100))

# remove outliers

loan_without_outlier <- loan_decimal %>% filter(annual_inc < 500000 & revol_bal < 300000)

# comparative boxplots

boxplot <- function(m){
    ggplot(loan_without_outlier, aes(x=!!as.name(m), y=as.factor(loan_status), fill=as.factor(loan_status))) +
    geom_boxplot() +
    labs(title = as.character(m), y = 'Loan Status') +
    theme(legend.title = element_blank())
}

numerics <- c('loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'int_rate', 'installment', 'annual_inc', 'dti', 'delinq_2yrs',
          'fico_range_low', 'fico_range_high', 'inq_last_6mths', 'open_acc', 'revol_bal', 'revol_util', 'total_acc', 'out_prn
          cp', 'out_prncp_inv', 'total_rec_late_fee', 'last_pymnt_amnt', 'issue_d_year', 'earliest_cr_line_year', 'last_pymnt
          _d_year', 'last_credit_pull_d_year')

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

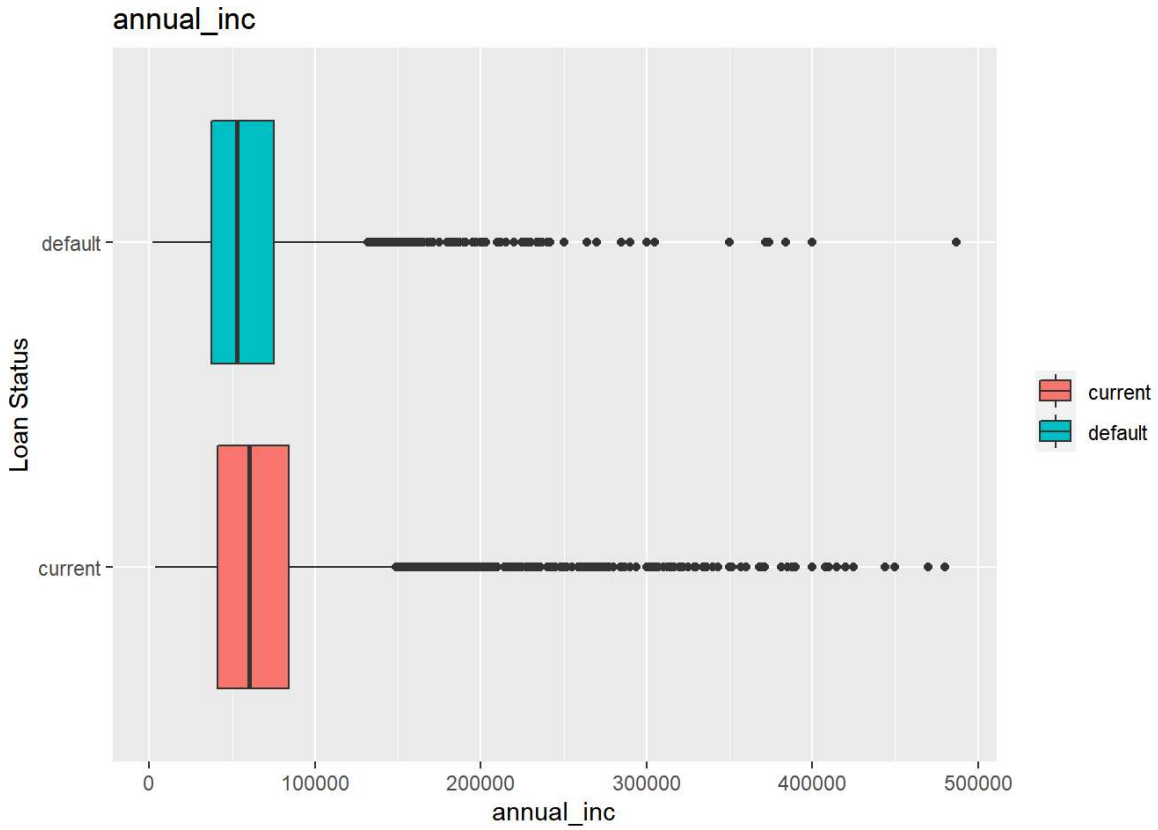

loan_amnt
default
current
Loan Status
0
10000
20000
30000
loan_amnt
current
default

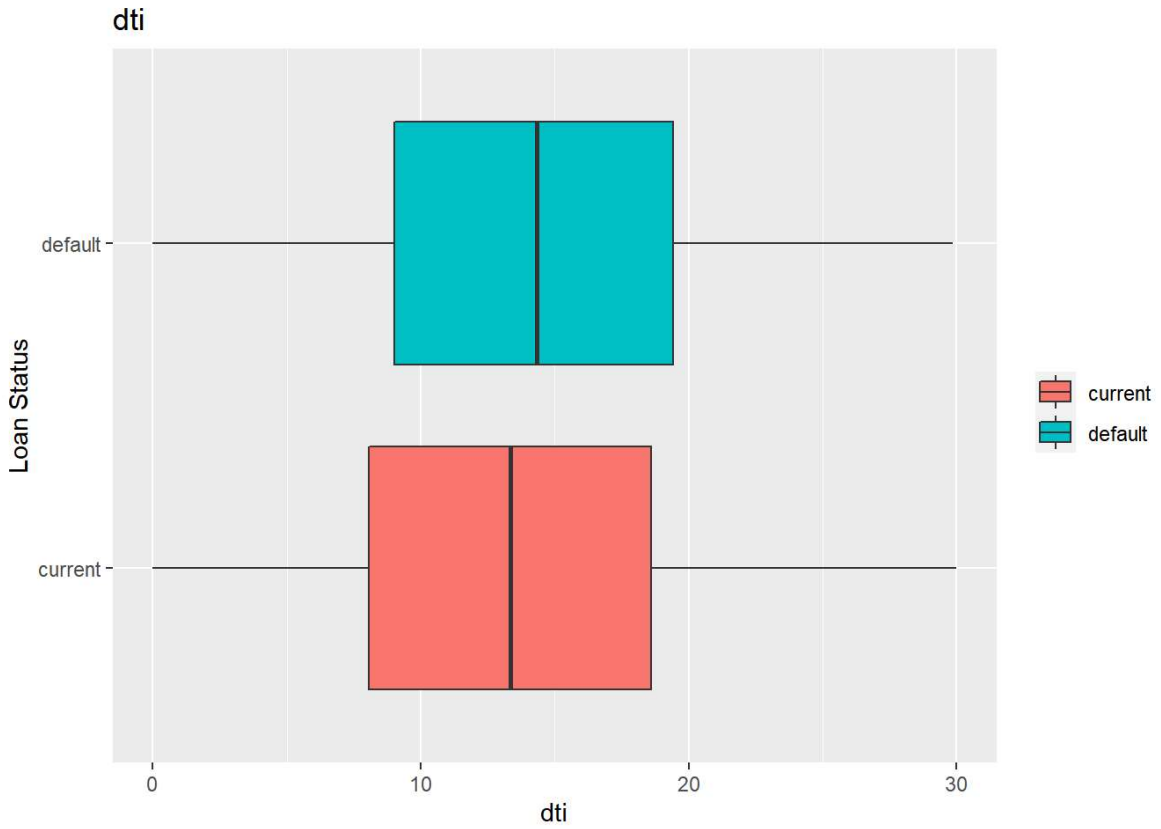

funded_amnt
default
current
Loan Status
0
10000
20000
30000
funded_amnt
current
default

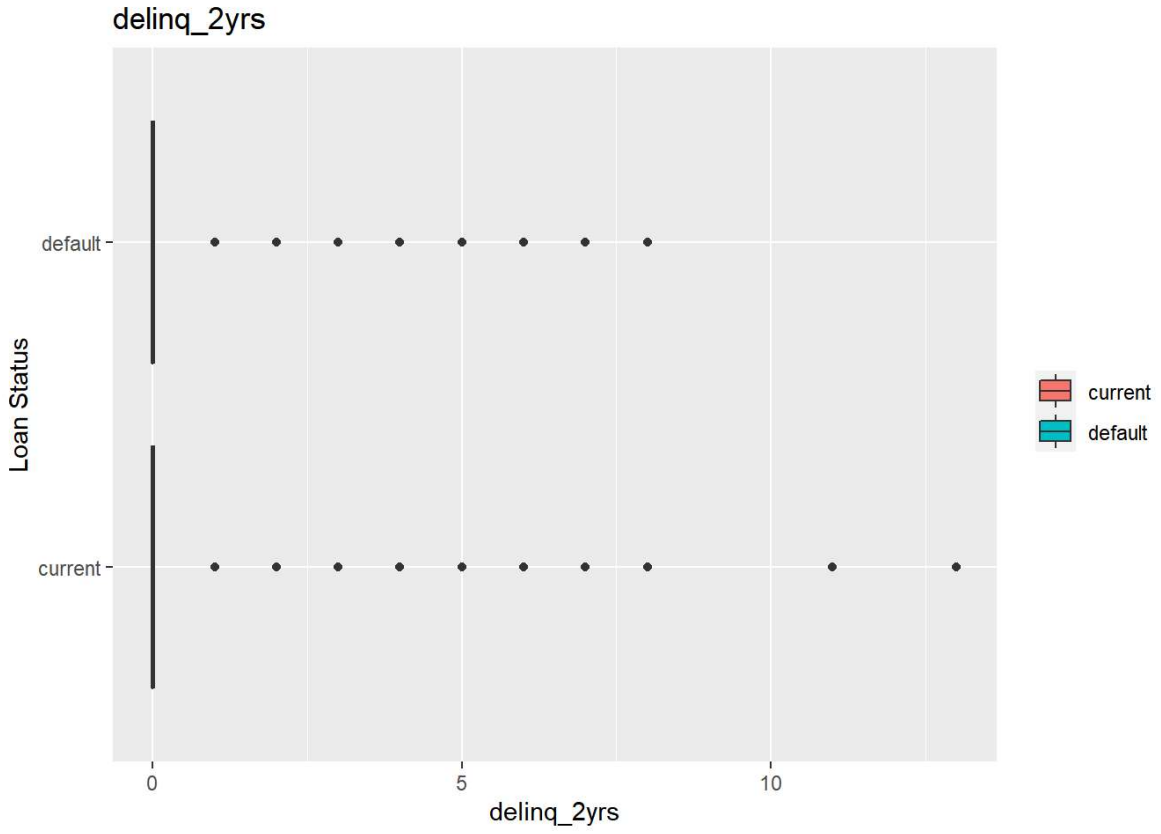

funded_amnt_inv
default
current
Loan Status
0
10000
20000
30000
funded_amnt_inv
current
default

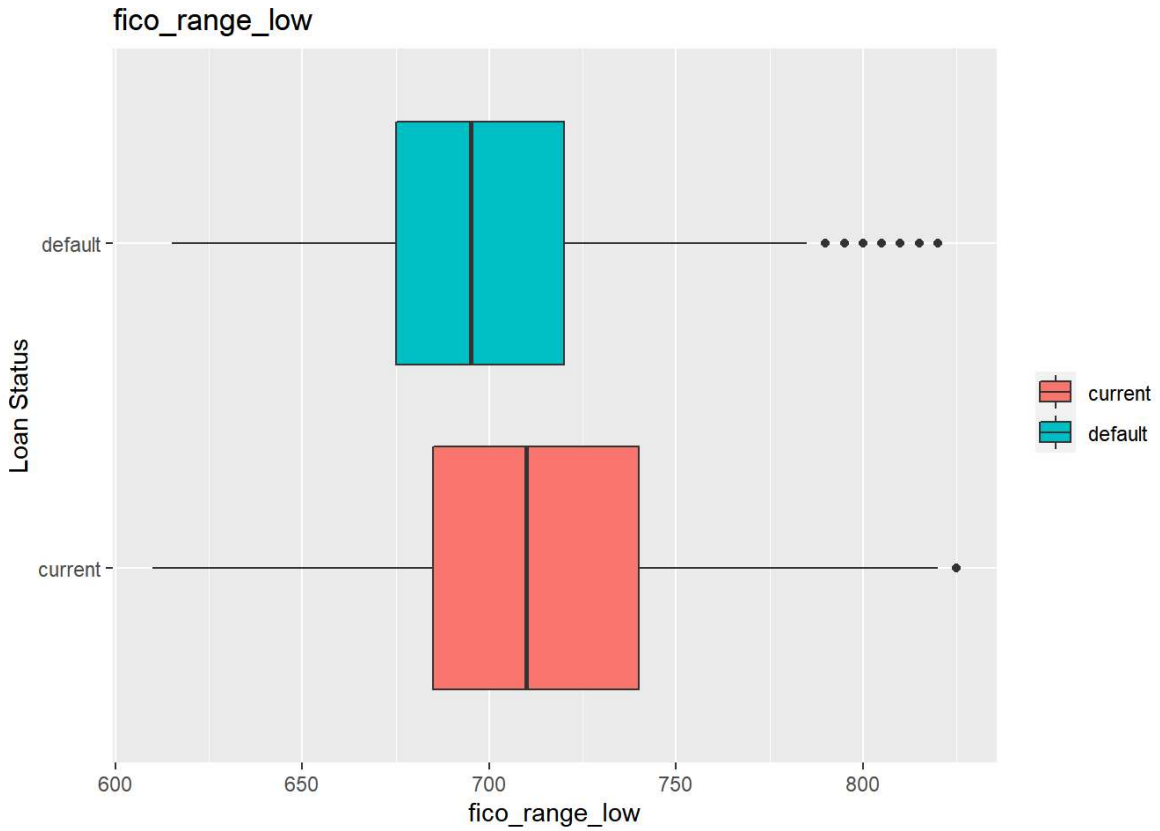

int_rate
default
current
Loan Status
0.05
0.10
0.15
0.20
0.25
int_rate
current
default

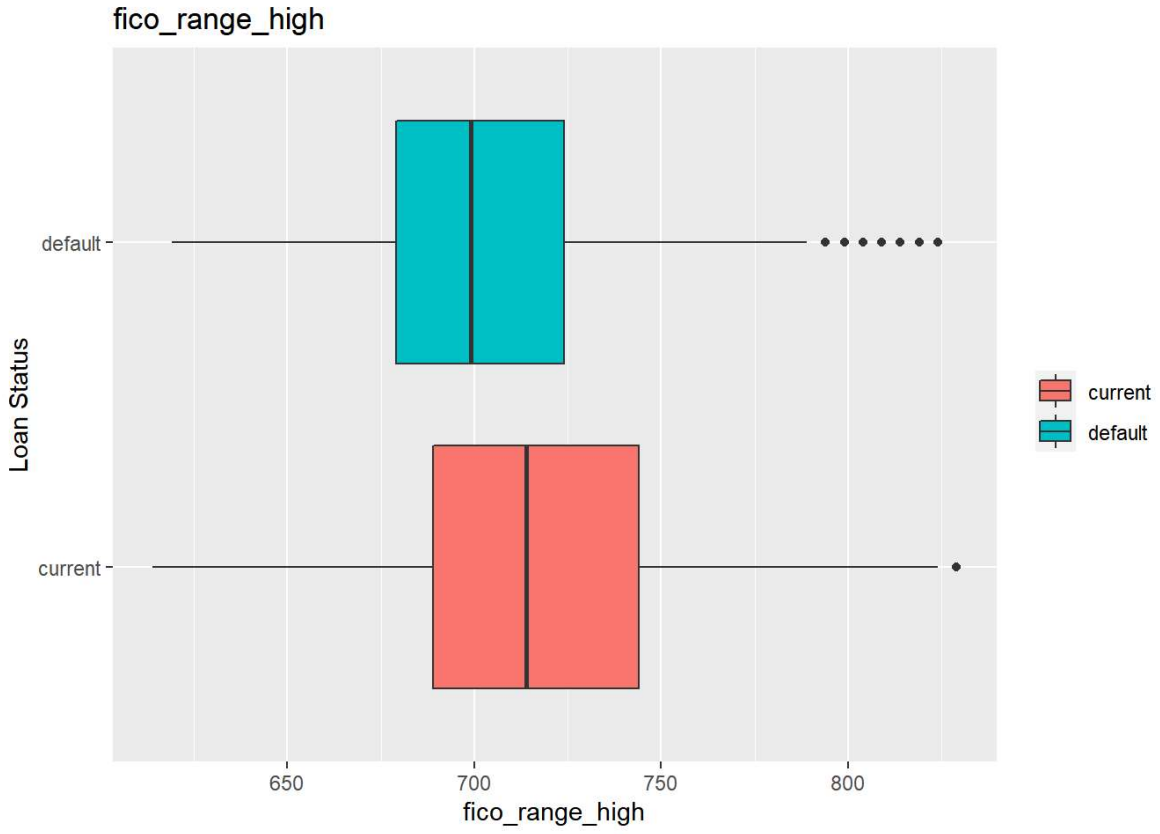

installment
default
current
Loan Status
0
500
1000
installment
current
default

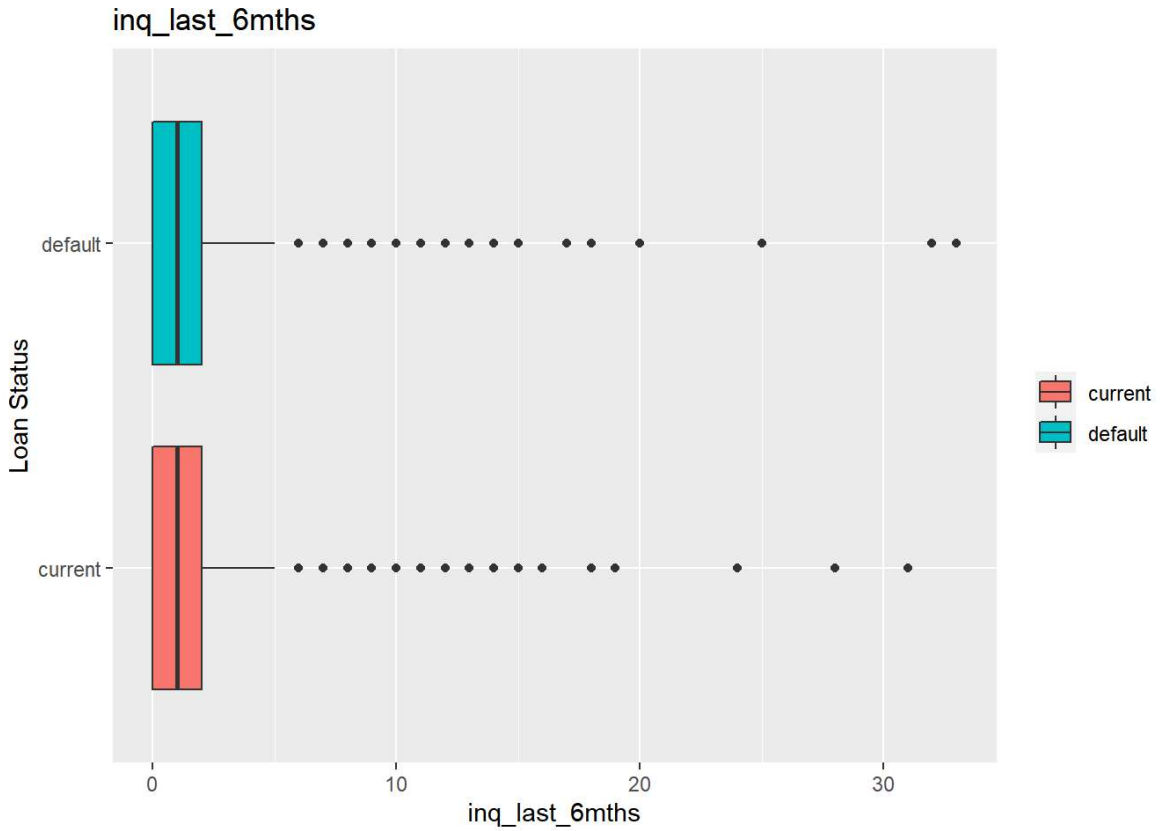

annual_inc
default
current
Loan Status
0
100000
200000
300000
400000
500000
annual_inc
current
default

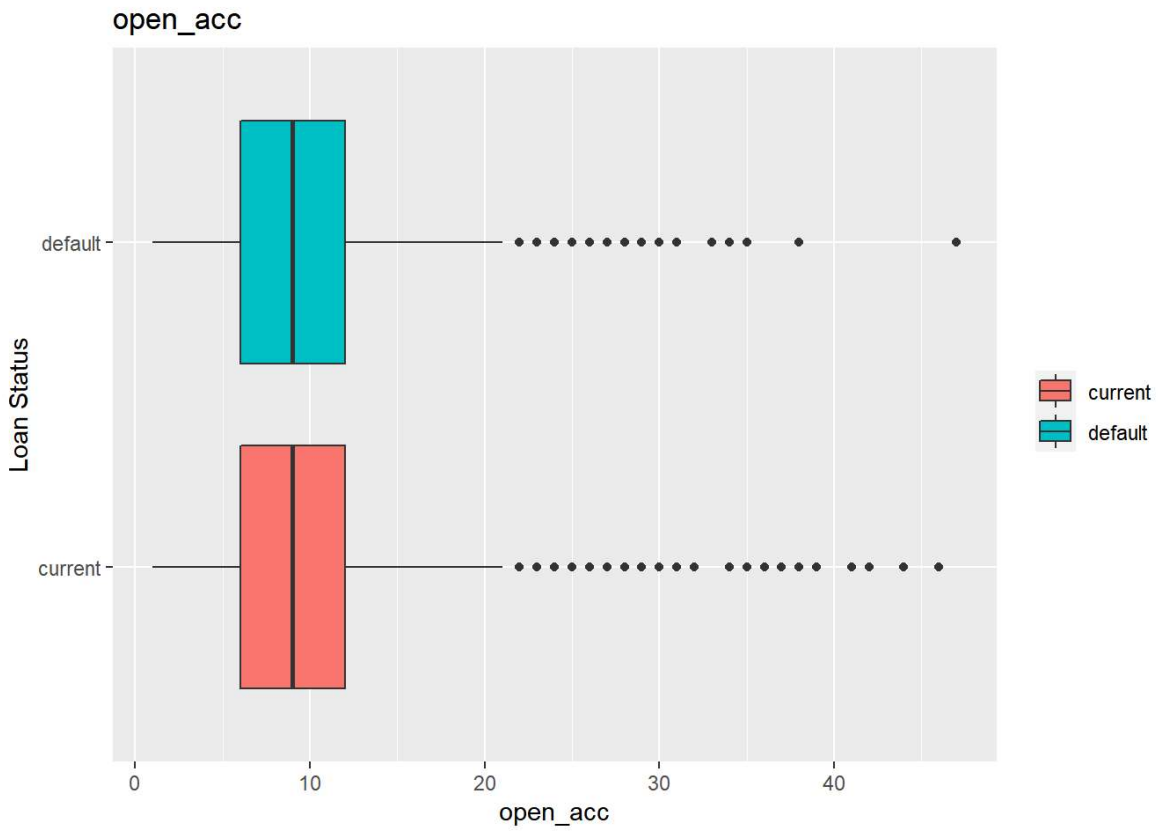

dti
default
current
Loan Status
0
10
20
30
dti
current
default

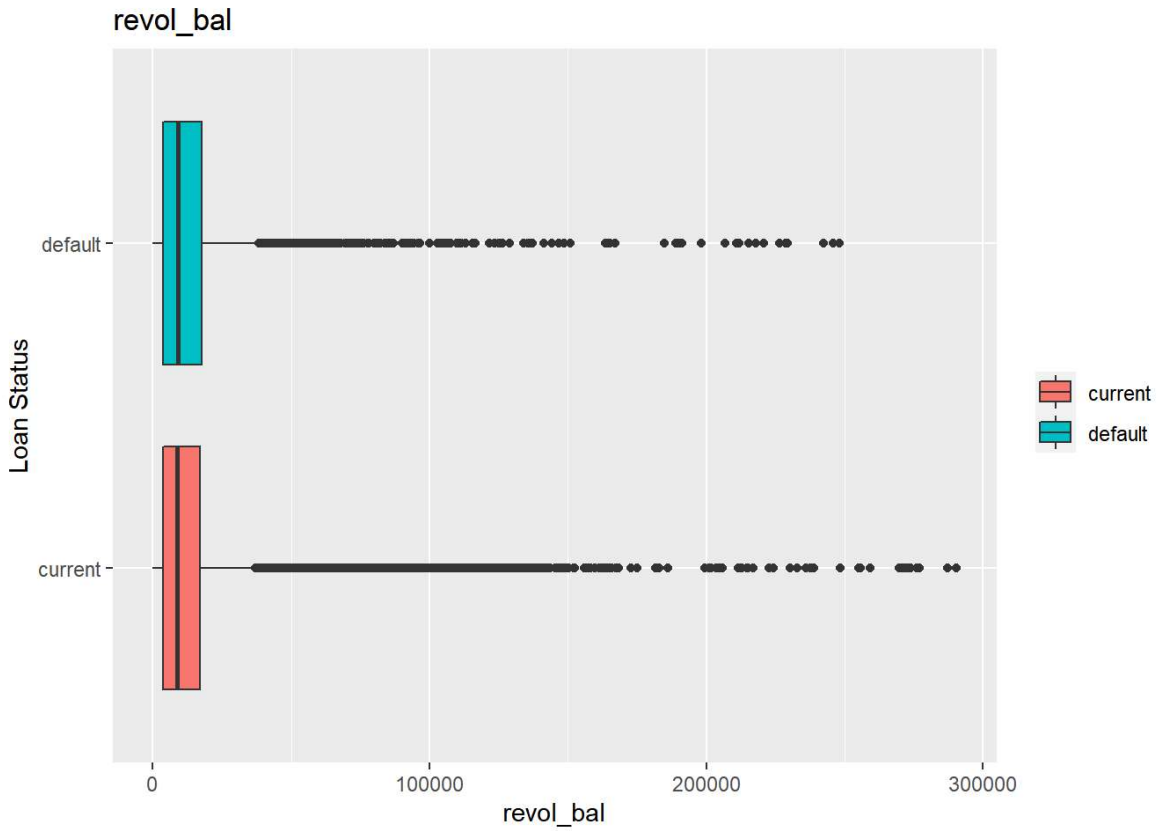

delinq_2yrs
default
current
Loan Status
0
5
10
delinq_2yrs
current
default

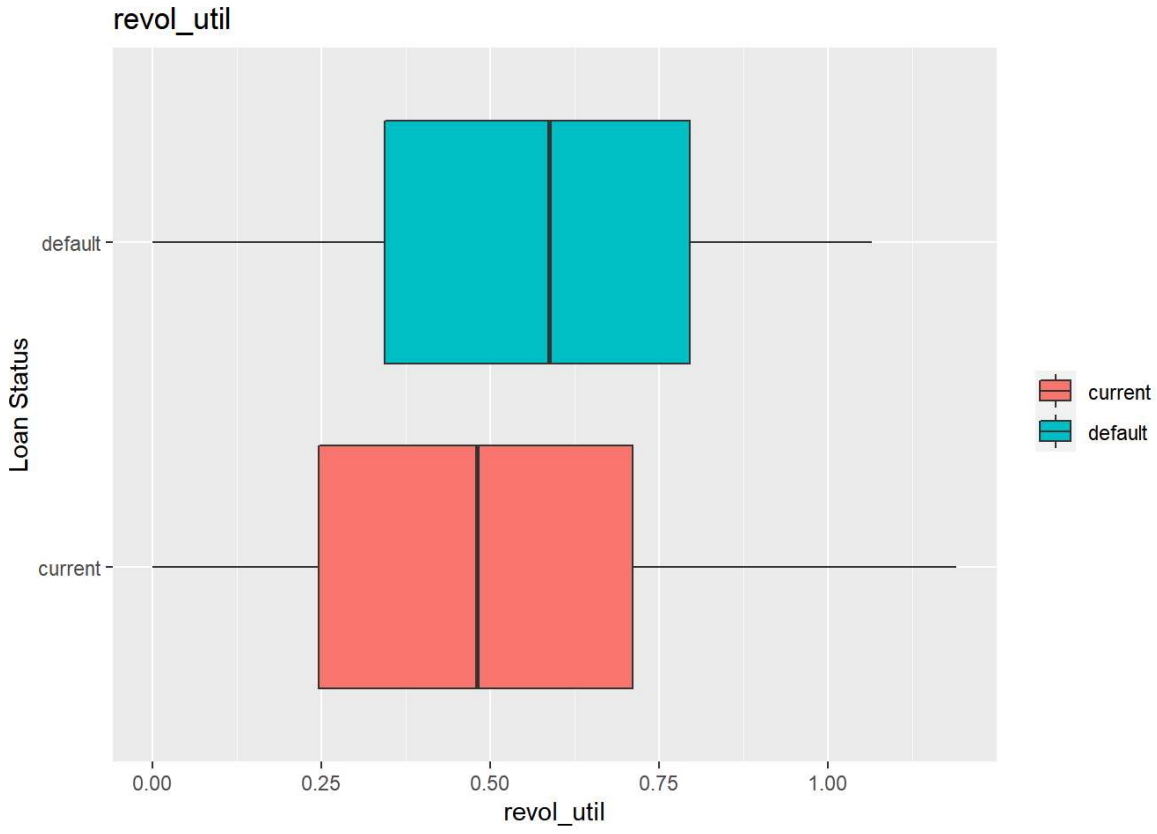

fico_range_low
default
current
Loan Status
600
650
700
750
800
fico_range_low
current
default

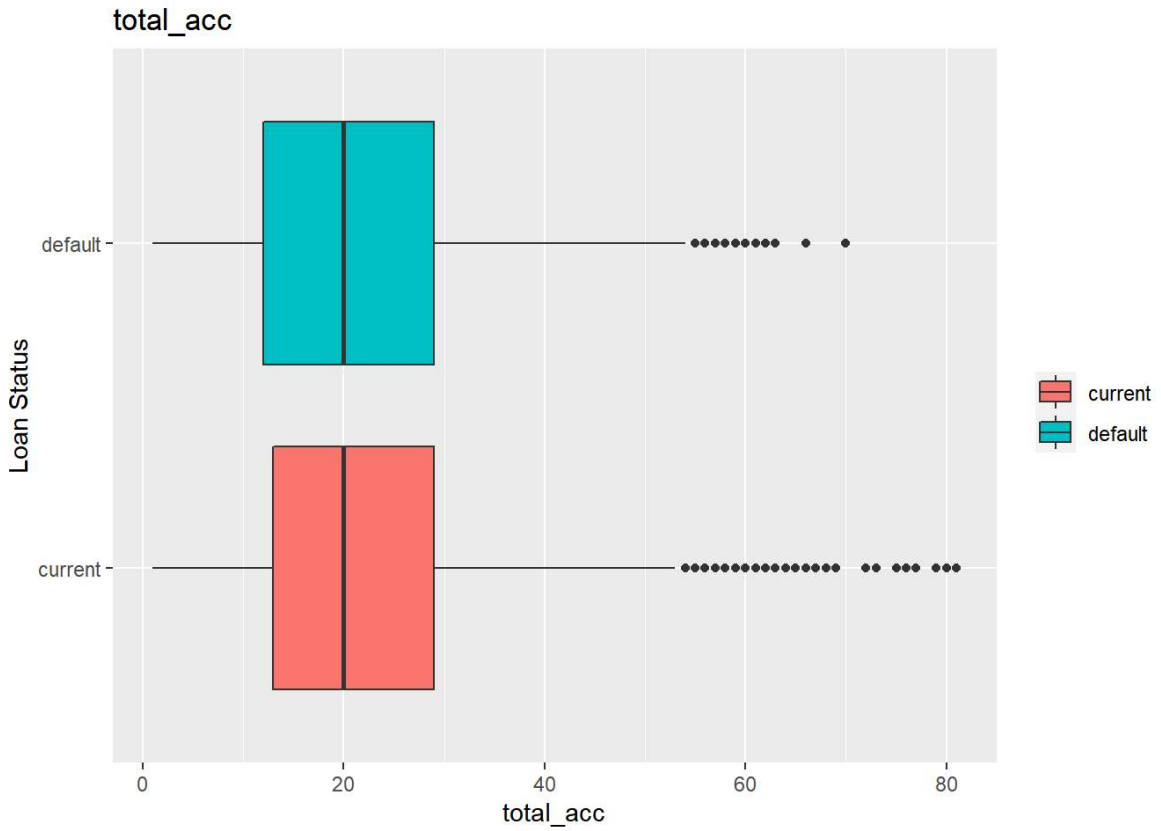

fico_range_high
default
current
Loan Status
650
700
750
800
fico_range_high
current
default

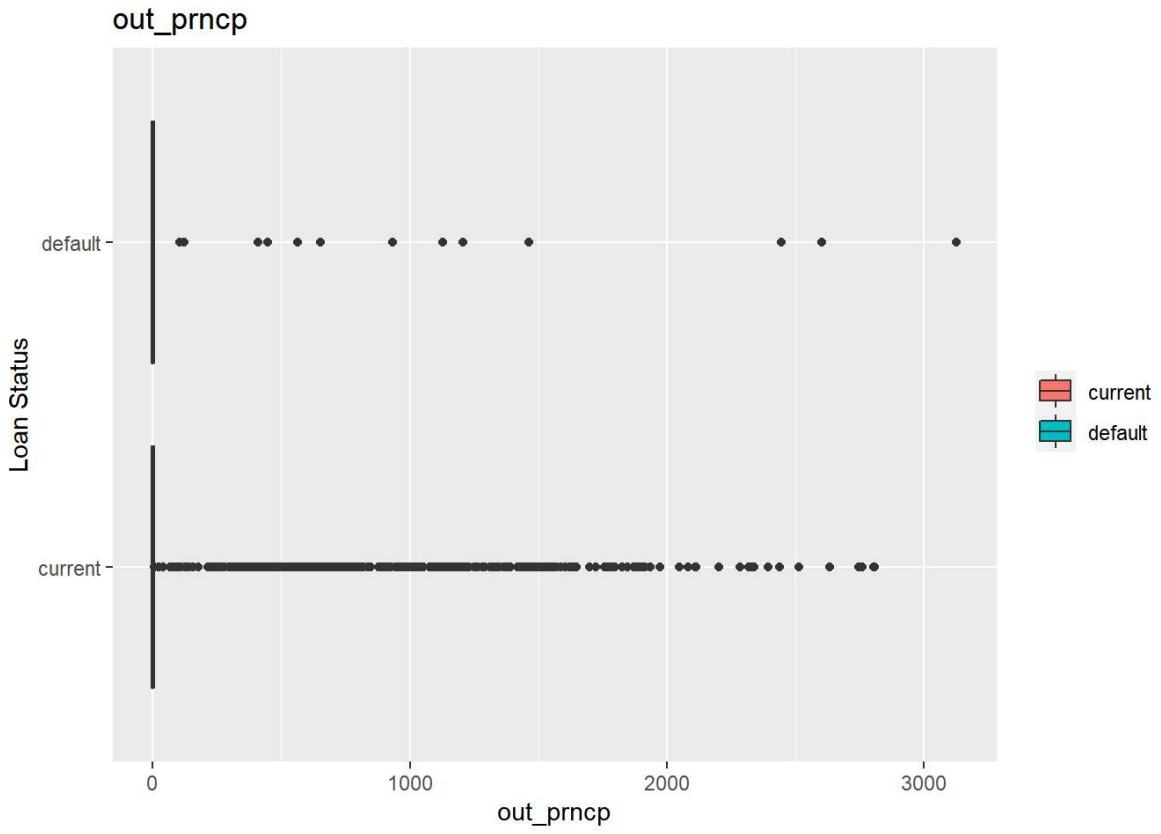

inq_last_6mths
default
current
Loan Status
0
10
20
30
inq_last_6mths
current
default

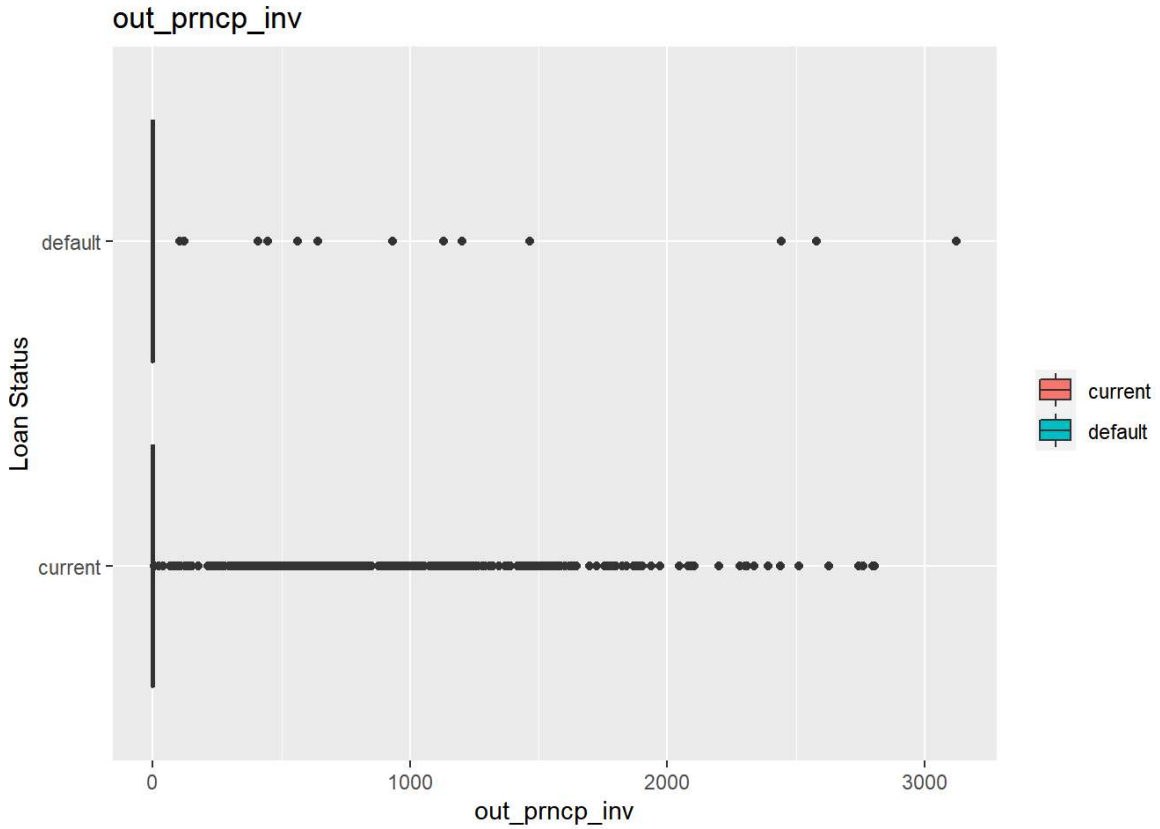

open_acc
default
current
Loan Status
0
10
20
30
40
open_acc
current
default

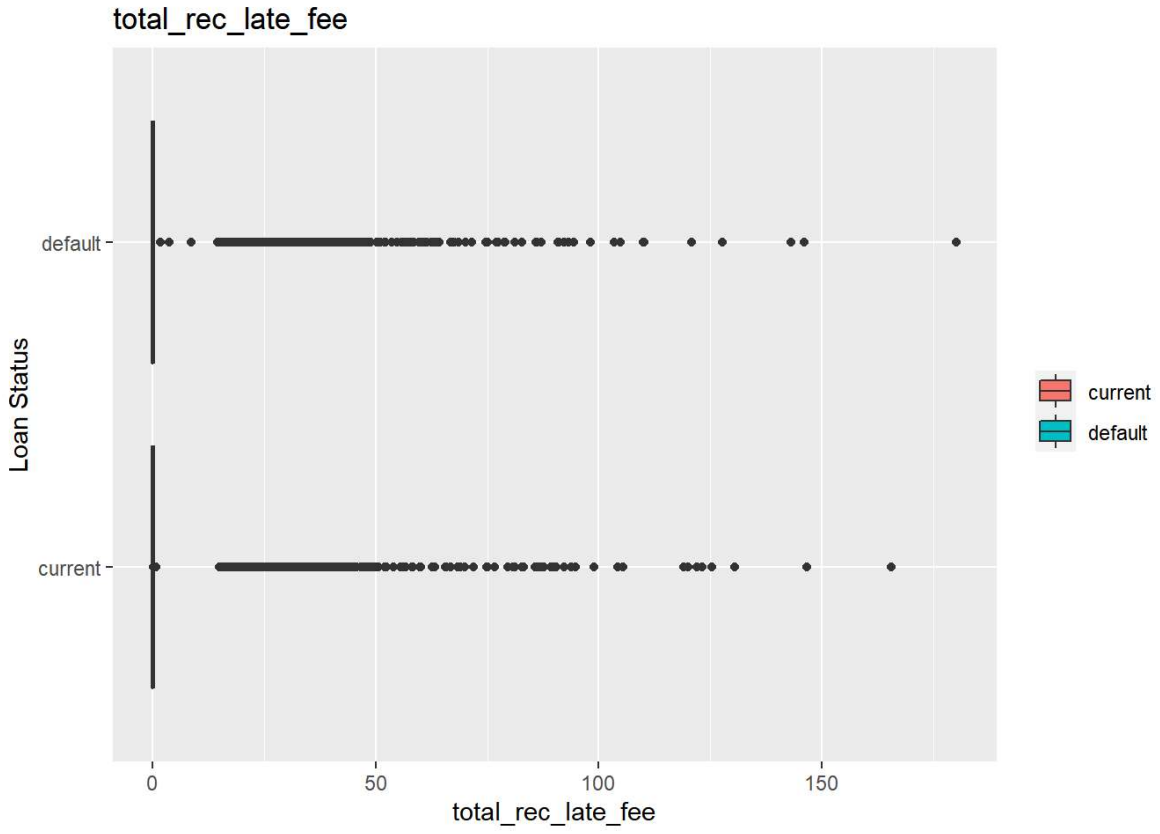

revol_bal
default
current
Loan Status
0
100000
200000
300000
revol_bal
current
default

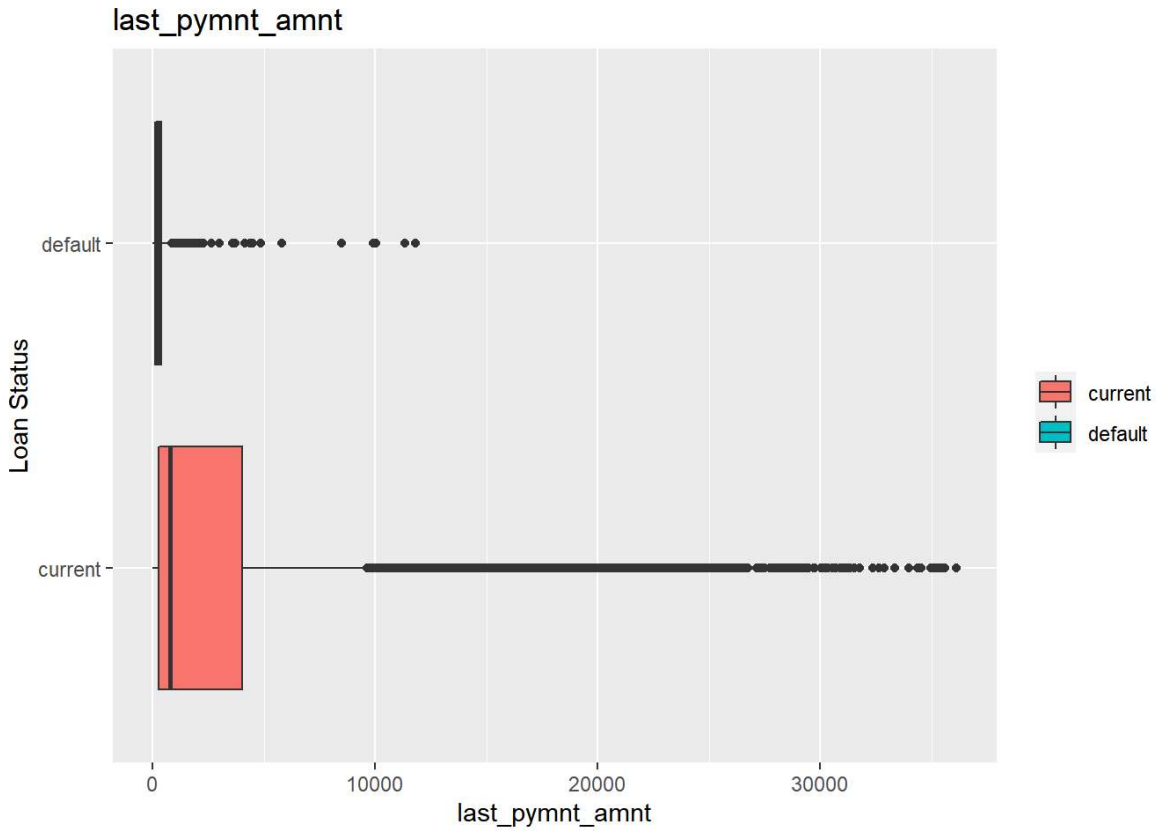

revol_util
default
current
Loan Status
0.00
0.25
0.50
0.75
1.00
revol_util
current
default

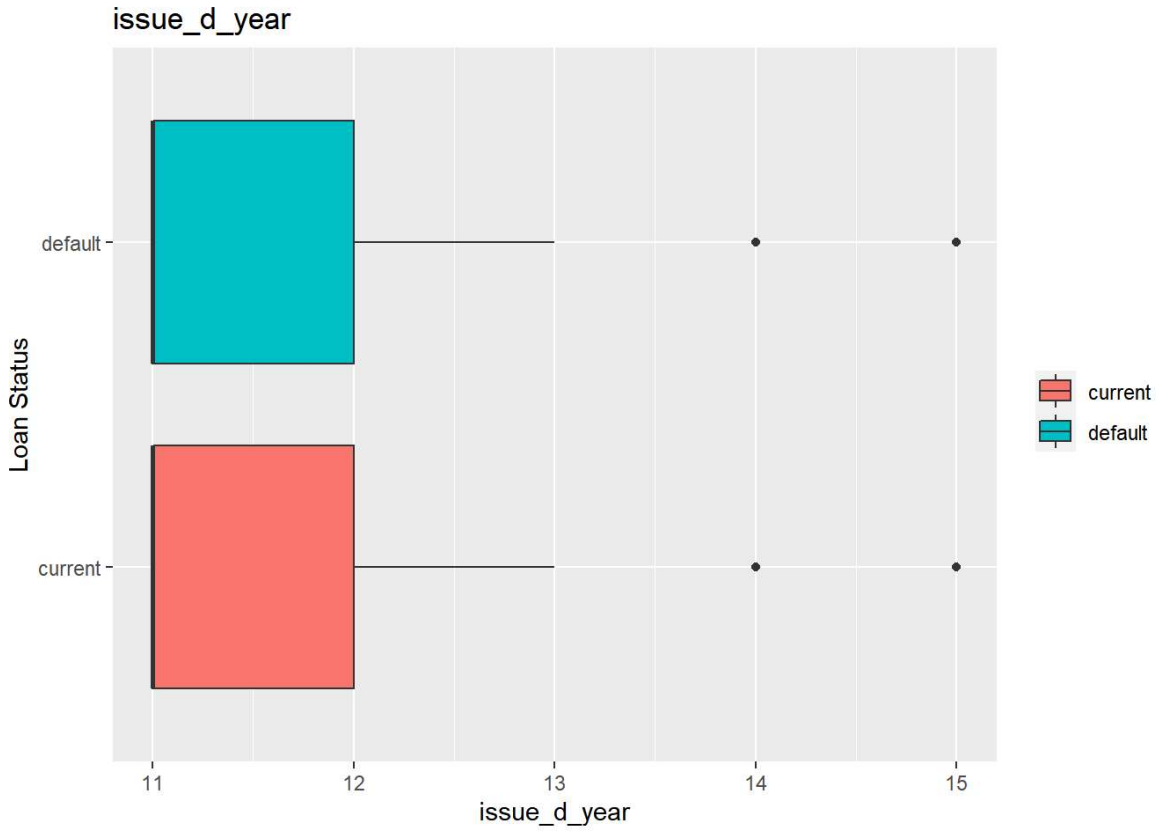

total_acc
default
current
Loan Status
0
20
40
60
80
total_acc
current
default

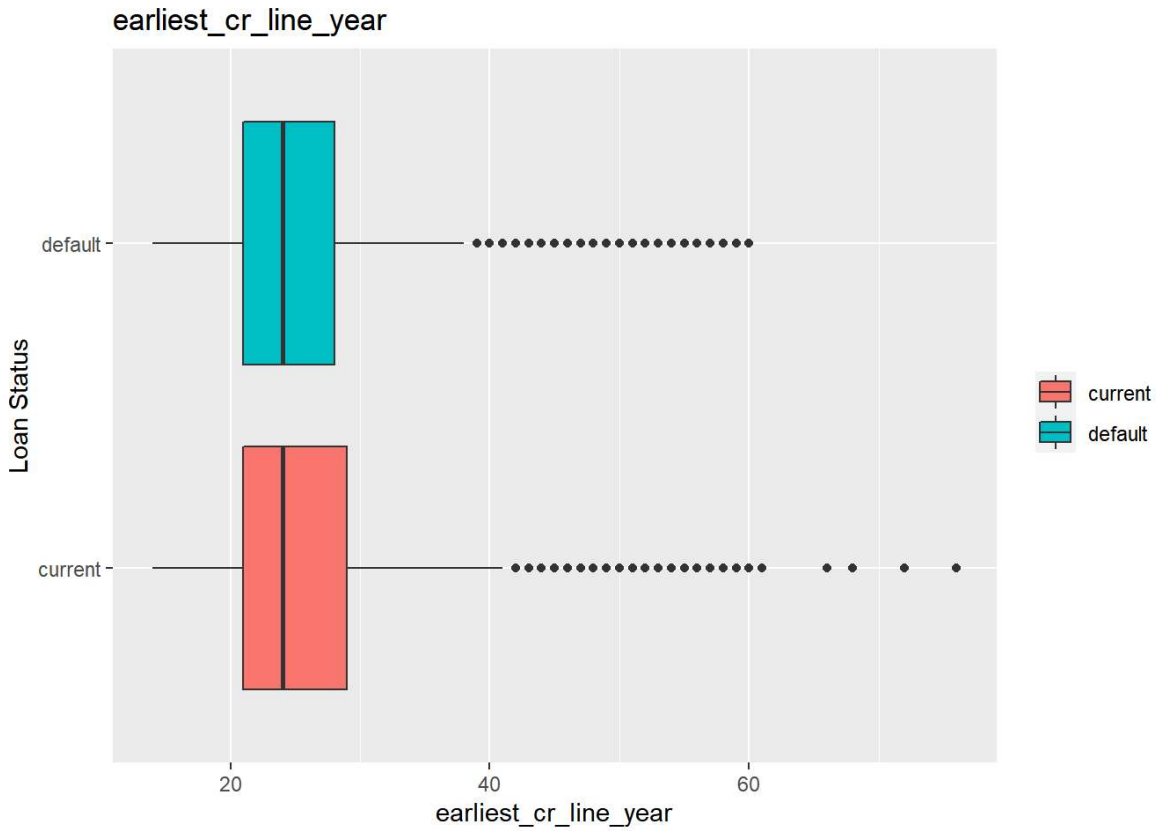

out_prncp
default
current
Loan Status
0
1000
2000
3000
out_prncp
current
default

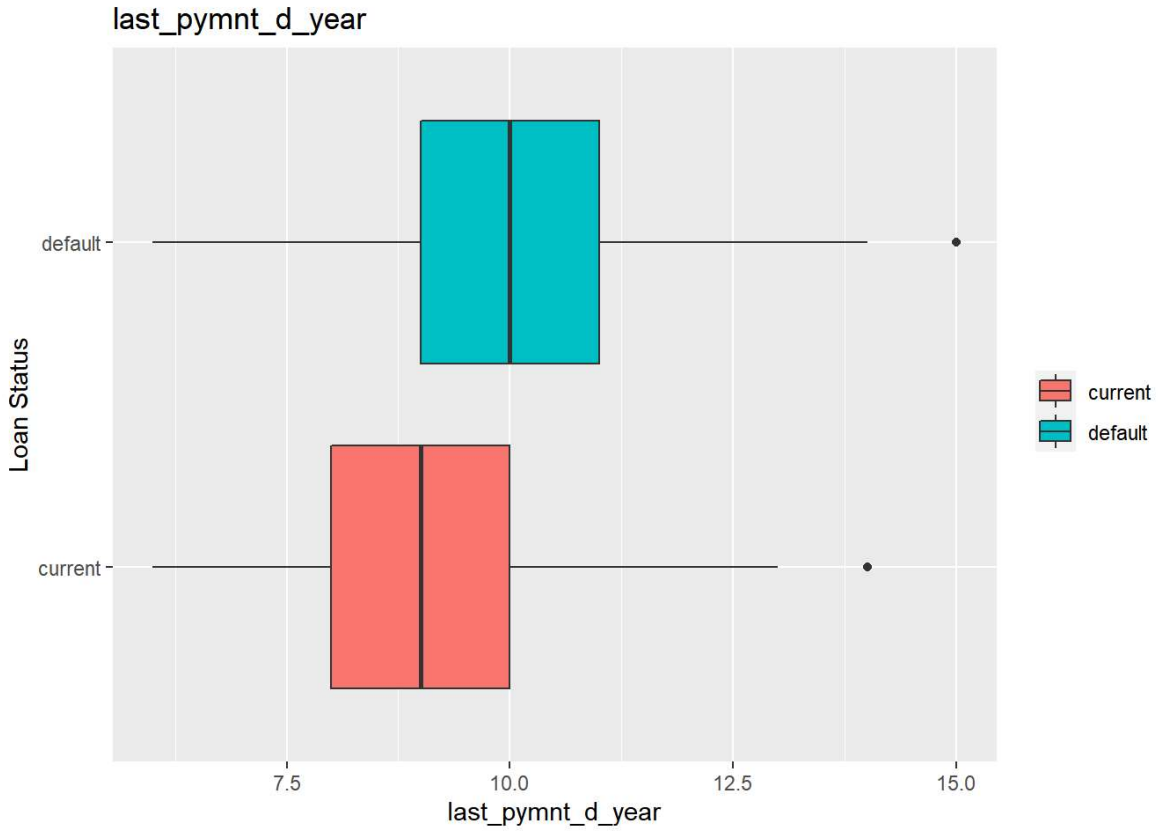

out_prncp_inv
default
current
Loan Status
0
1000
2000
3000
out_prncp_inv
current
default

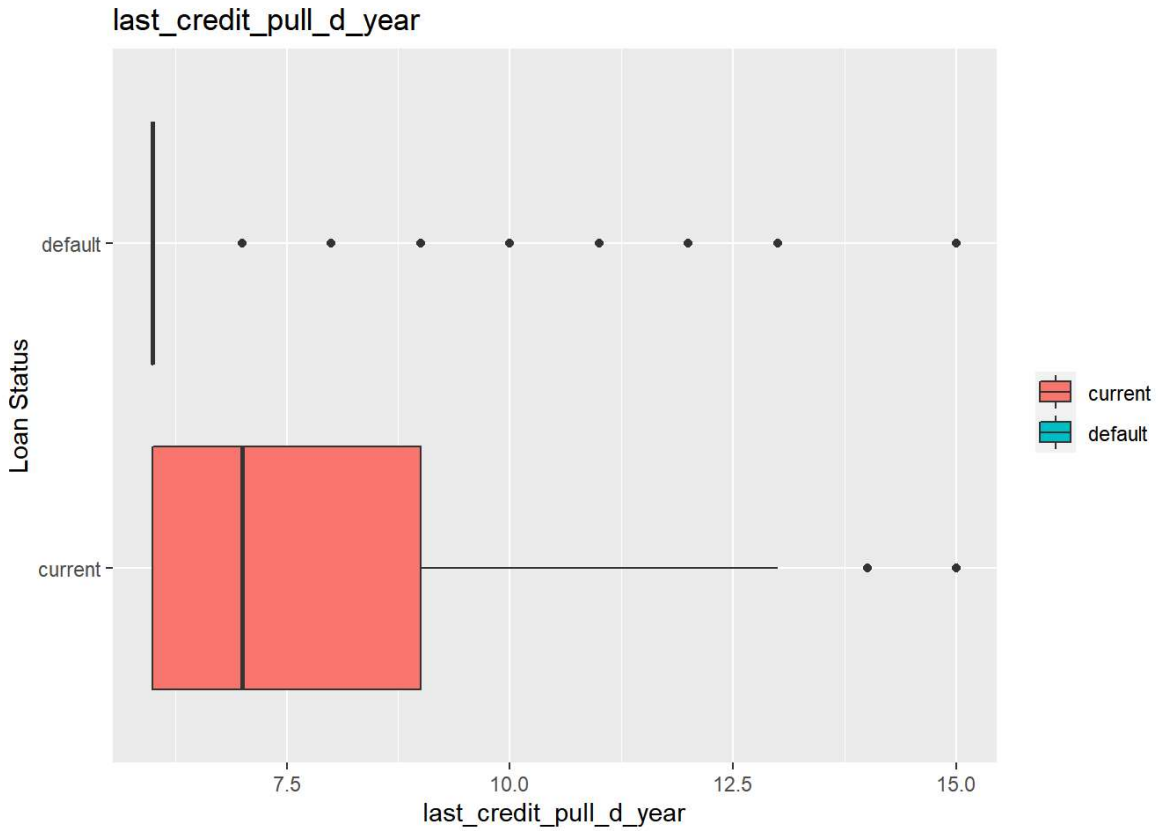

total_rec_late_fee
default
current
Loan Status
0
50
100
150
total_rec_late_fee
current
default


last_pymnt_amnt
default
current
Loan Status
0
10000
20000
30000
last_pymnt_amnt
current
default

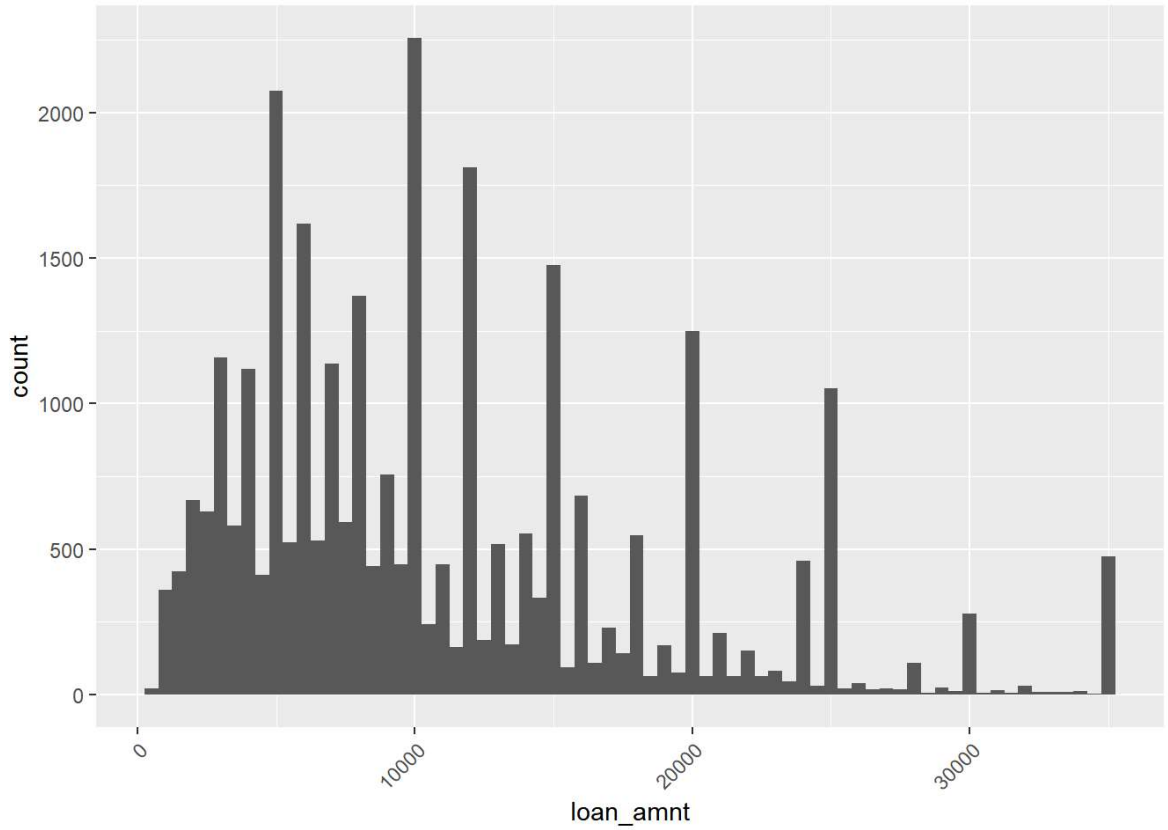

issue_d_year
default
current
Loan Status
11
12
13
14
15
issue_d_year
current
default


earliest_cr_line_year
default
current
Loan Status
20
40
60
earliest_cr_line_year
current
default

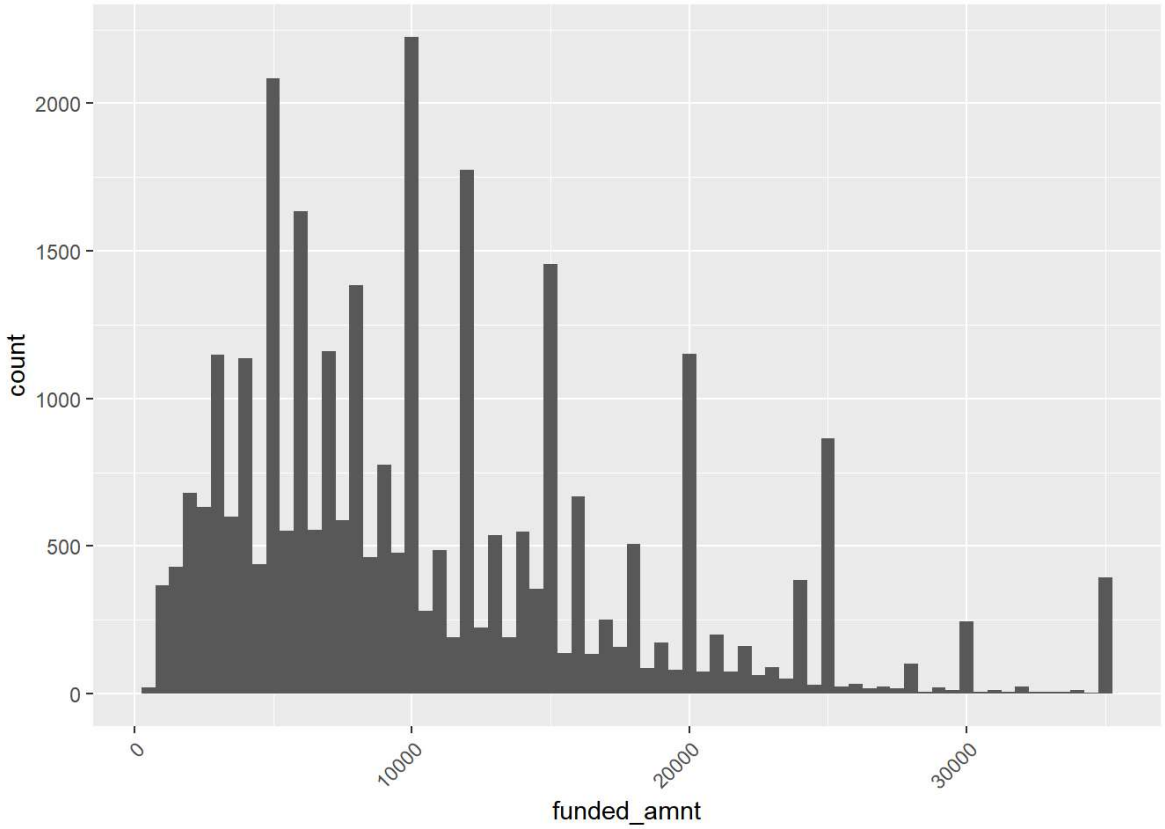

last_pymnt_d_year
default
current
Loan Status
7.5
10.0
12.5
15.0
last_pymnt_d_year
current
default



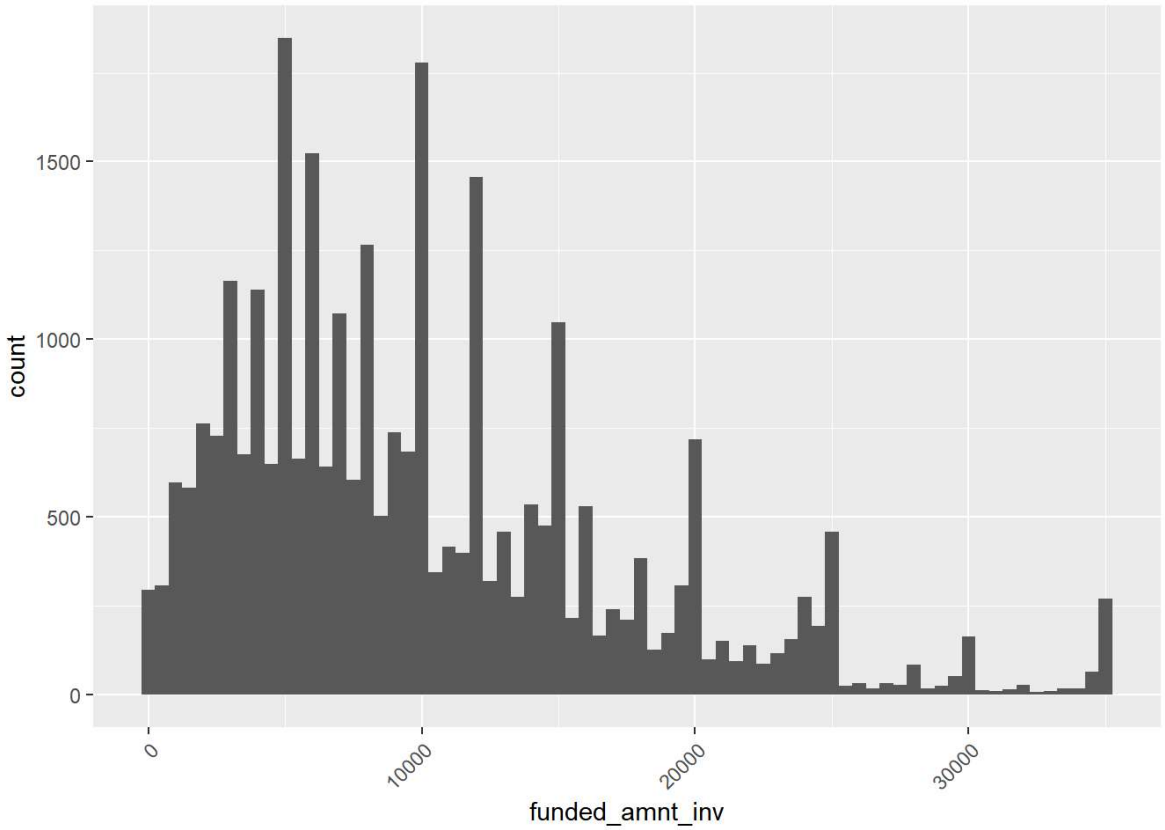

```
loan_without_outlier %>% ggplot(aes(x=loan_amnt)) + geom_histogram(binwidth=500) + theme(axis.text.x=element_text(angle=45, h
        just=1))
```



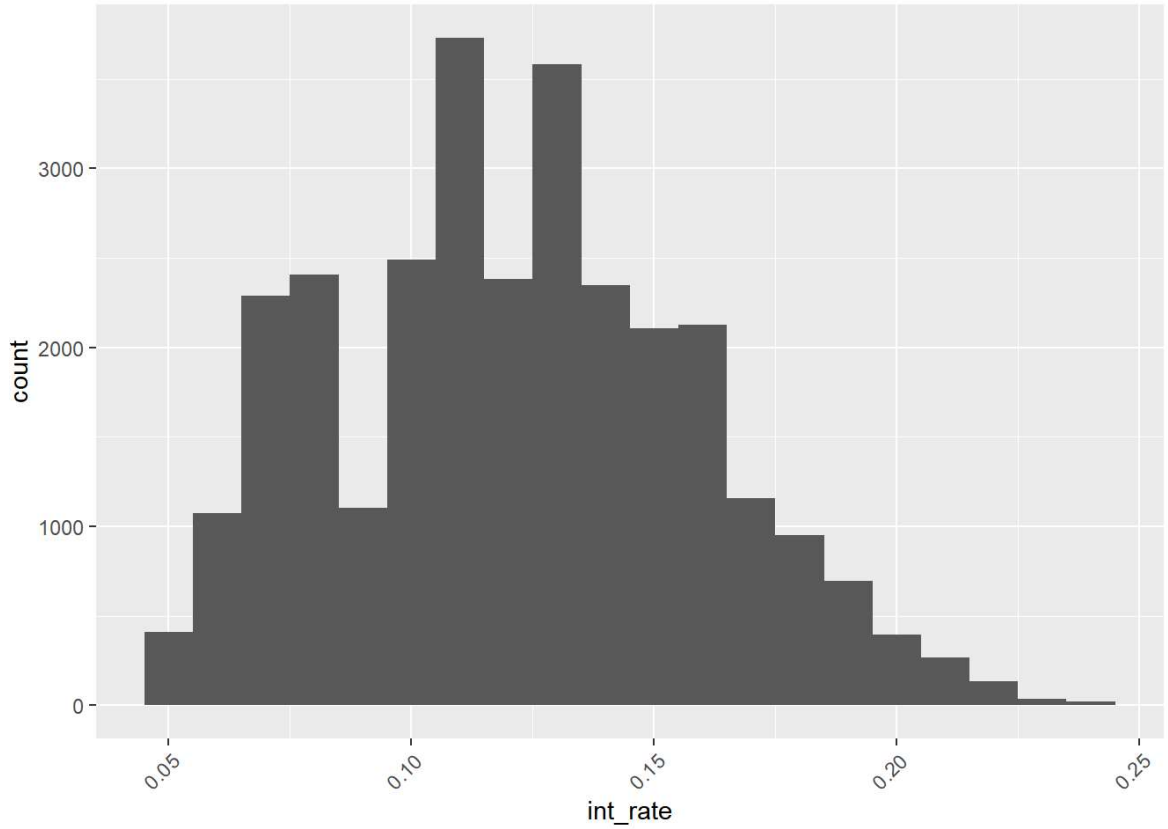

```
loan_without_outlier %>% ggplot(aes(x=funded_amnt)) + geom_histogram(binwidth=500) + theme(axis.text.x=element_text(angle=45,
        hjust=1))
```



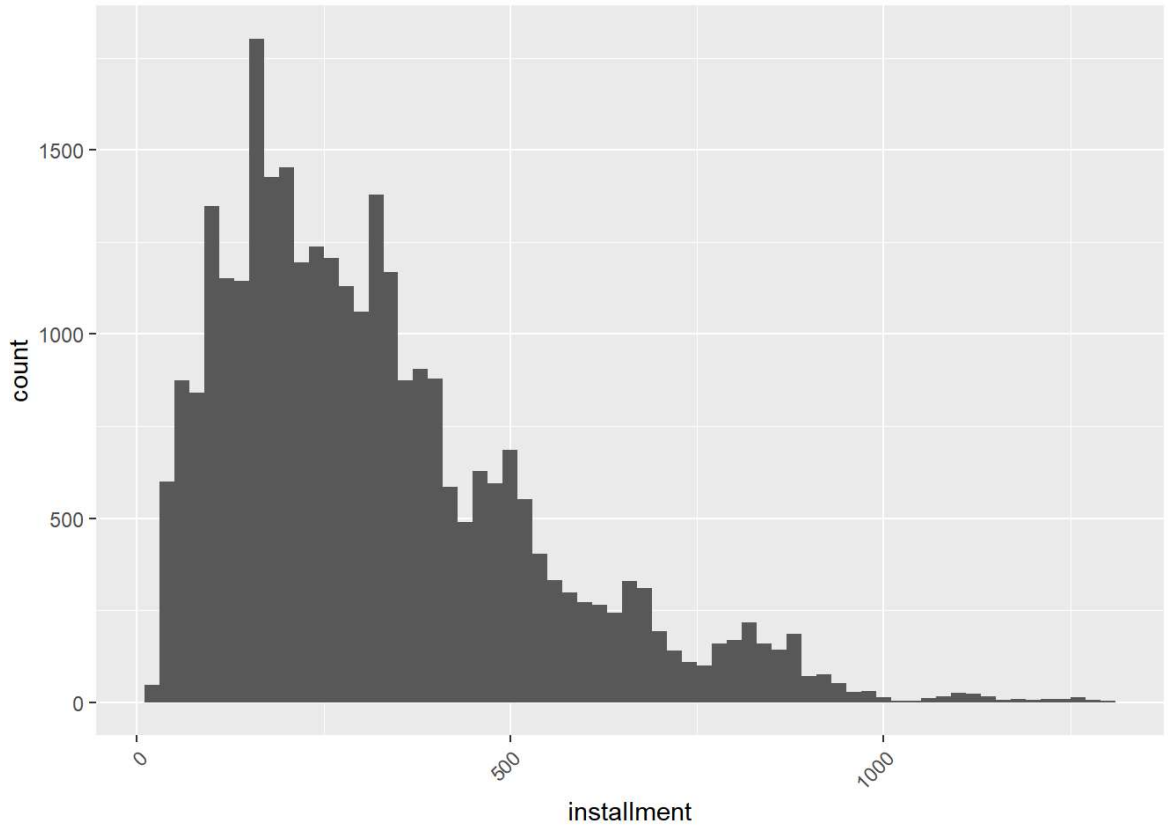

```
loan_without_outlier %>% ggplot(aes(x=funded_amnt_inv)) + geom_histogram(binwidth=500) + theme(axis.text.x=element_text(angle
        =45, hjust=1))
```



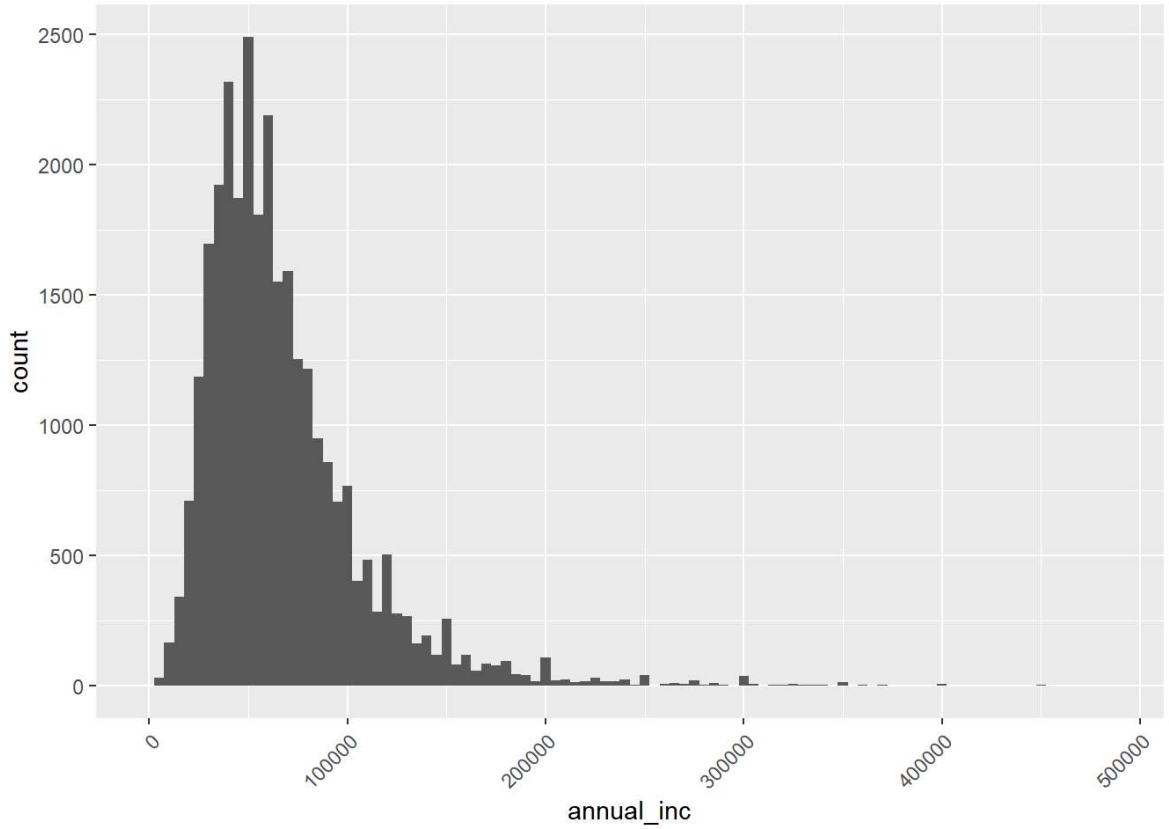

```
loan_without_outlier %>% ggplot(aes(x=int_rate)) + geom_histogram(binwidth=0.01) + theme(axis.text.x=element_text(angle=45, h
        just=1))
```



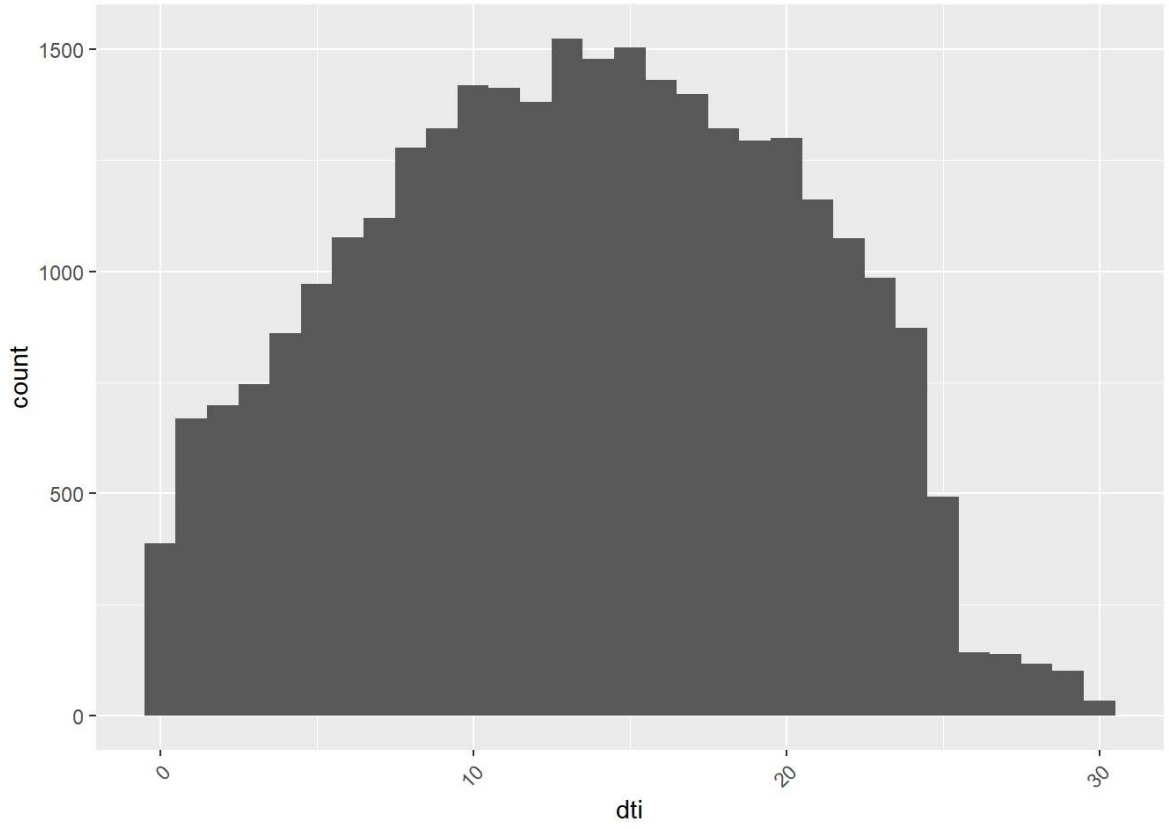

```
loan_without_outlier %>% ggplot(aes(x=installment)) + geom_histogram(binwidth=20) + theme(axis.text.x=element_text(angle=45,
          hjust=1))
```



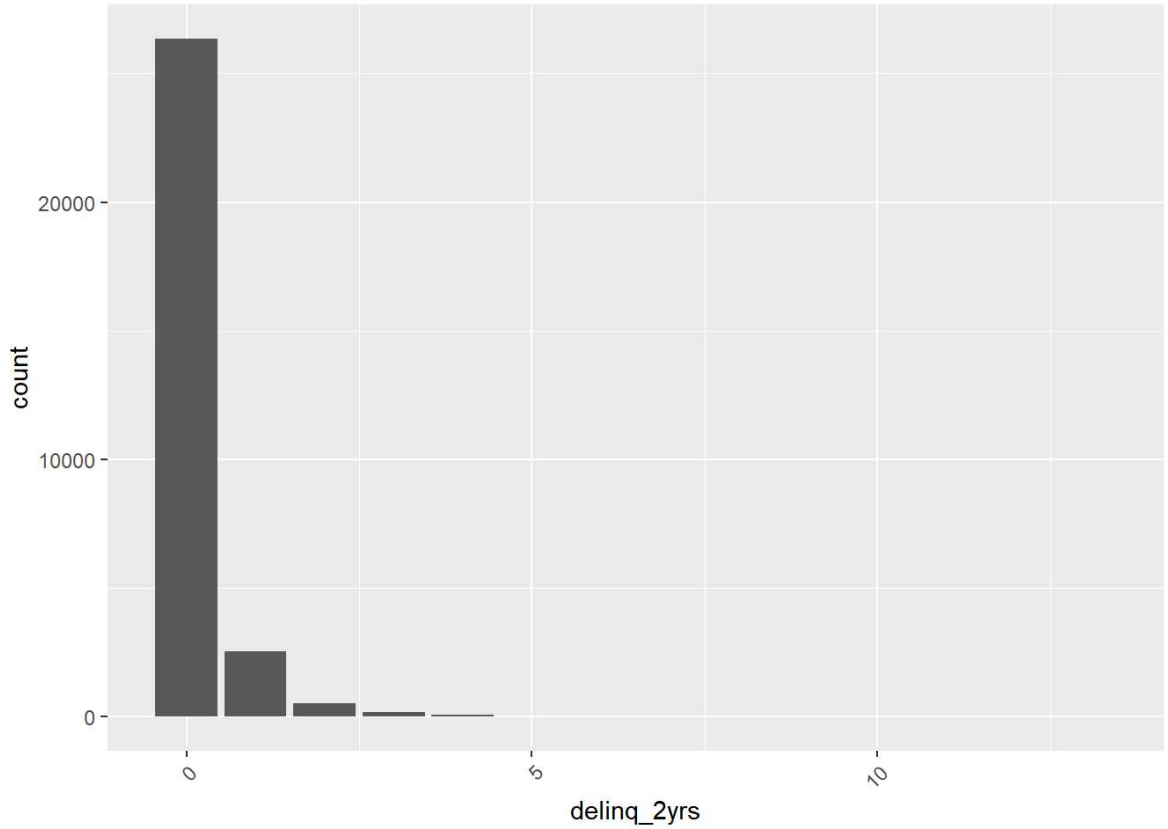

```
loan_without_outlier %>% ggplot(aes(x=annual_inc)) + geom_histogram(binwidth=5000) + theme(axis.text.x=element_text(angle=45,
          hjust=1))
```



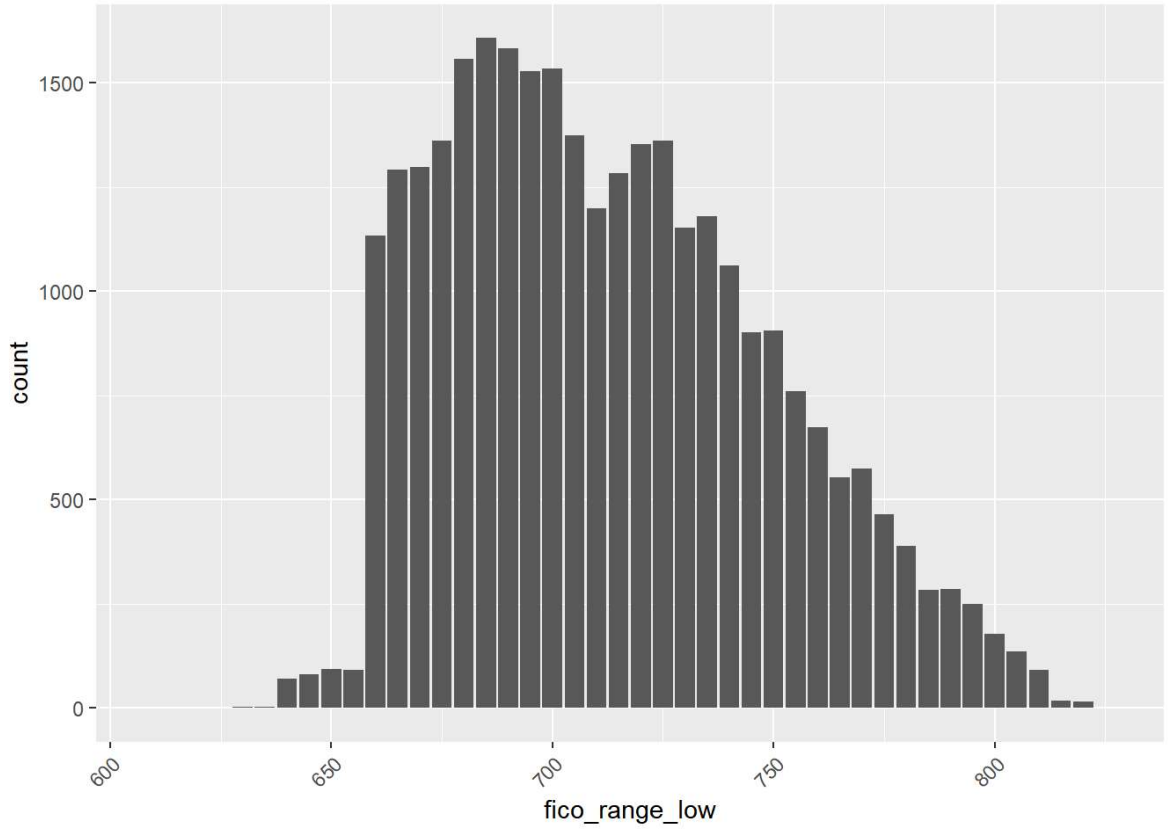

```
loan_without_outlier %>% ggplot(aes(x=dti)) + geom_histogram(binwidth=1) + theme(axis.text.x=element_text(angle=45, hjust=1))
```



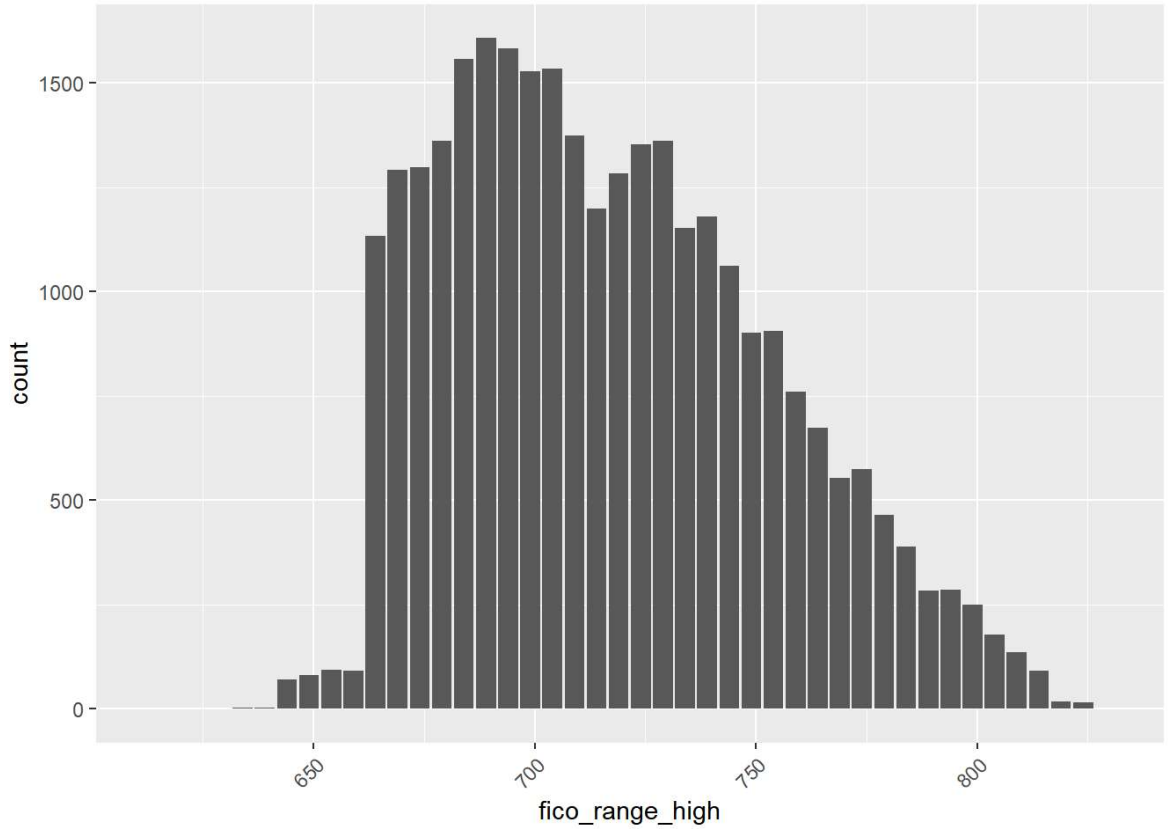

```
loan_without_outlier %>% ggplot(aes(x=delinq_2yrs)) + geom_bar() + theme(axis.text.x=element_text(angle=45, hjust=1))
```



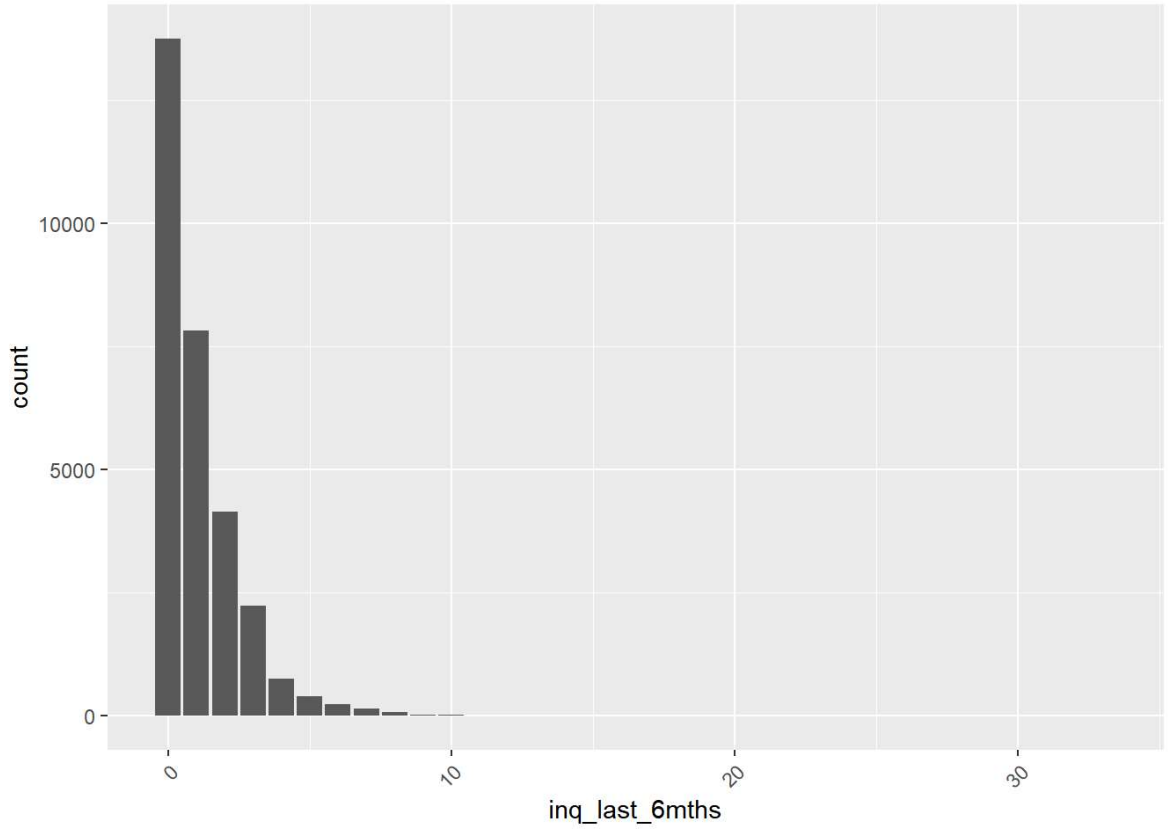

```
loan_without_outlier %>% ggplot(aes(x=fico_range_low)) + geom_bar() + theme(axis.text.x=element_text(angle=45, hjust=1))
```



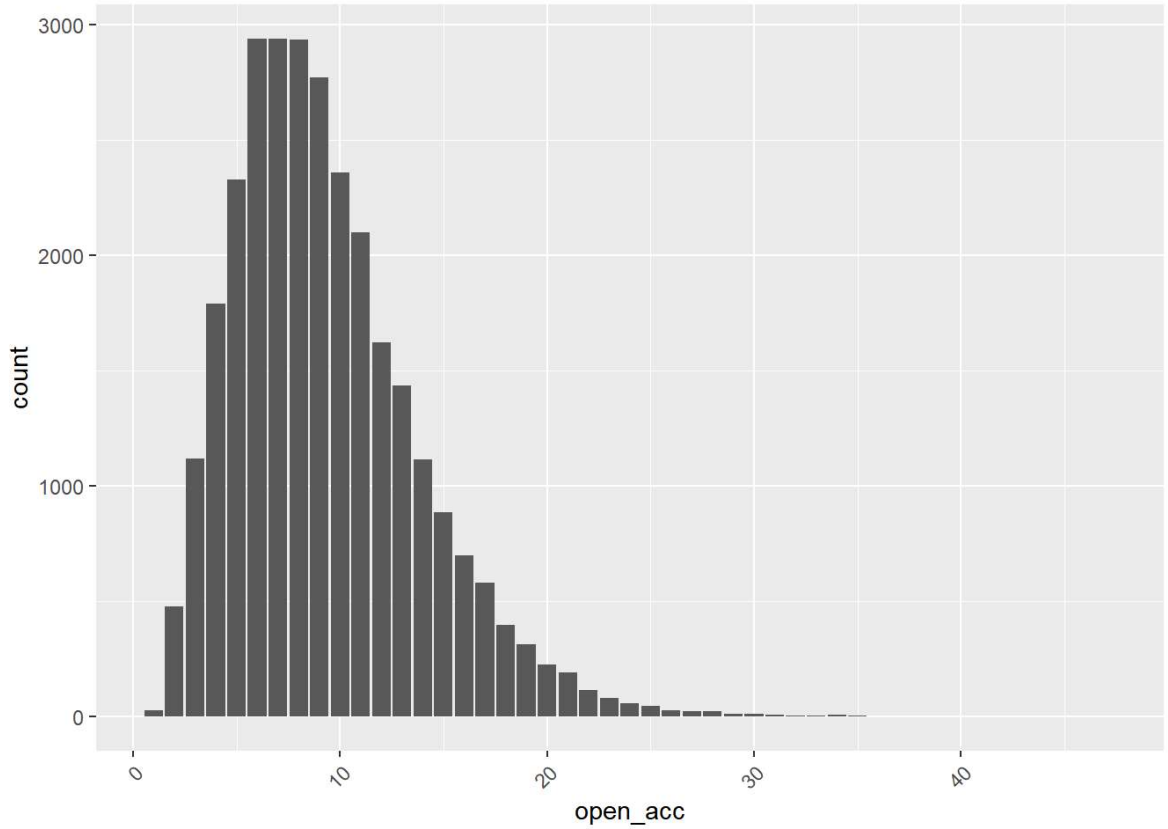

```
loan_without_outlier %>% ggplot(aes(x=fico_range_high)) + geom_bar() + theme(axis.text.x=element_text(angle=45, hjust=1))
```



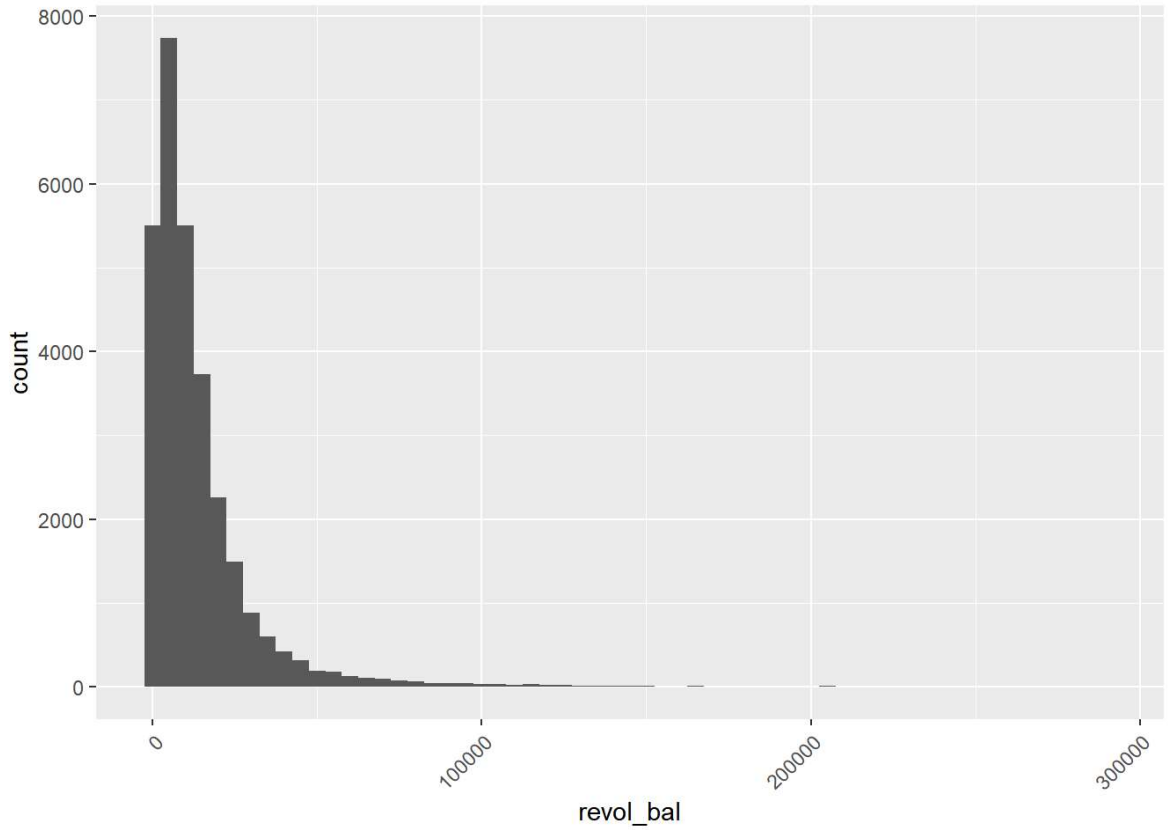

```
loan_without_outlier %>% ggplot(aes(x=inq_last_6mths)) + geom_bar() + theme(axis.text.x=element_text(angle=45, hjust=1))
```



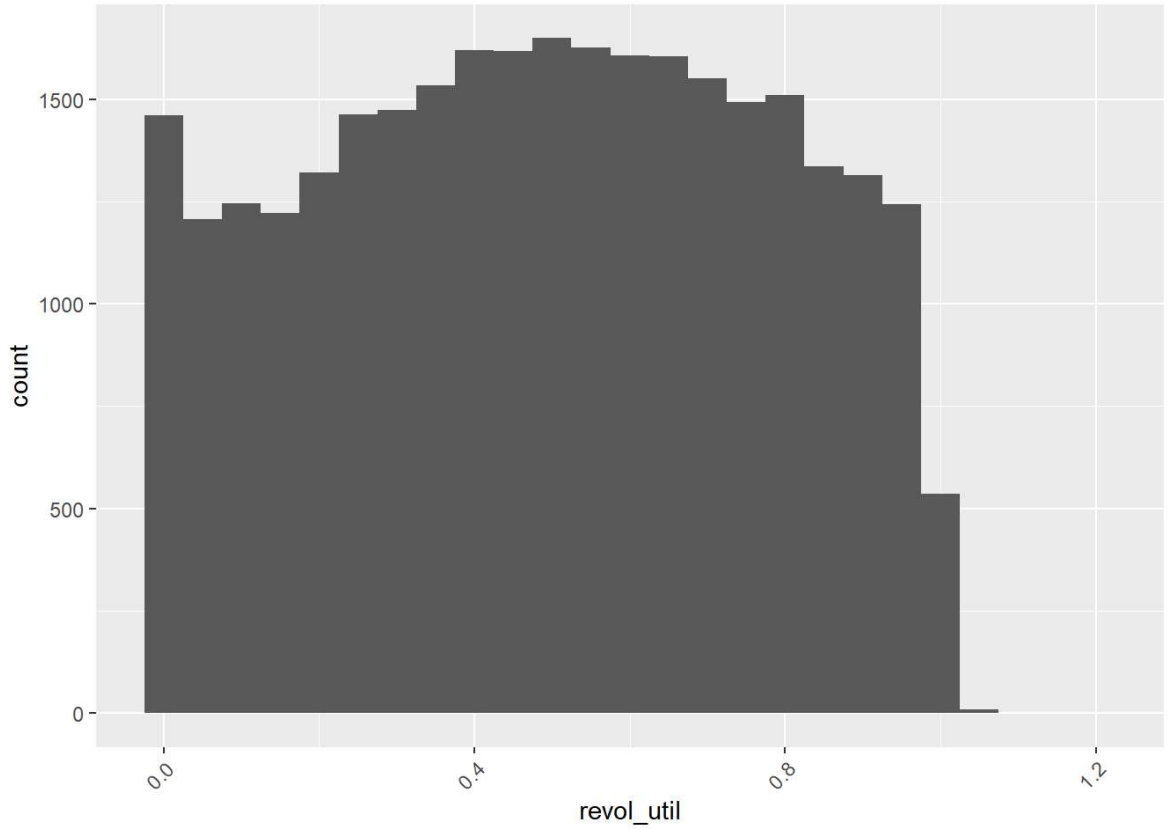

```
loan_without_outlier %>% ggplot(aes(x=open_acc)) + geom_bar() + theme(axis.text.x=element_text(angle=45, hjust=1))
```



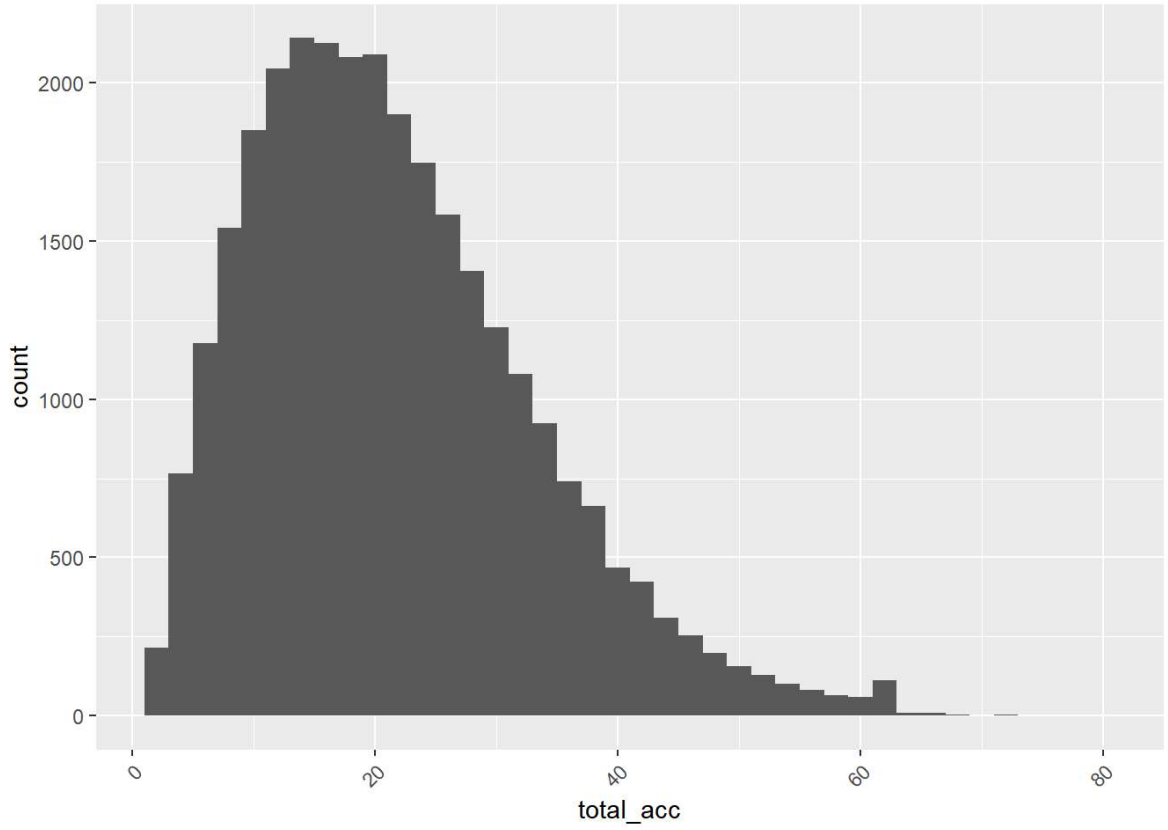

```
loan_without_outlier %>% ggplot(aes(x=revol_bal)) + geom_histogram(binwidth=5000) + theme(axis.text.x=element_text(angle=45,
        hjust=1))
```



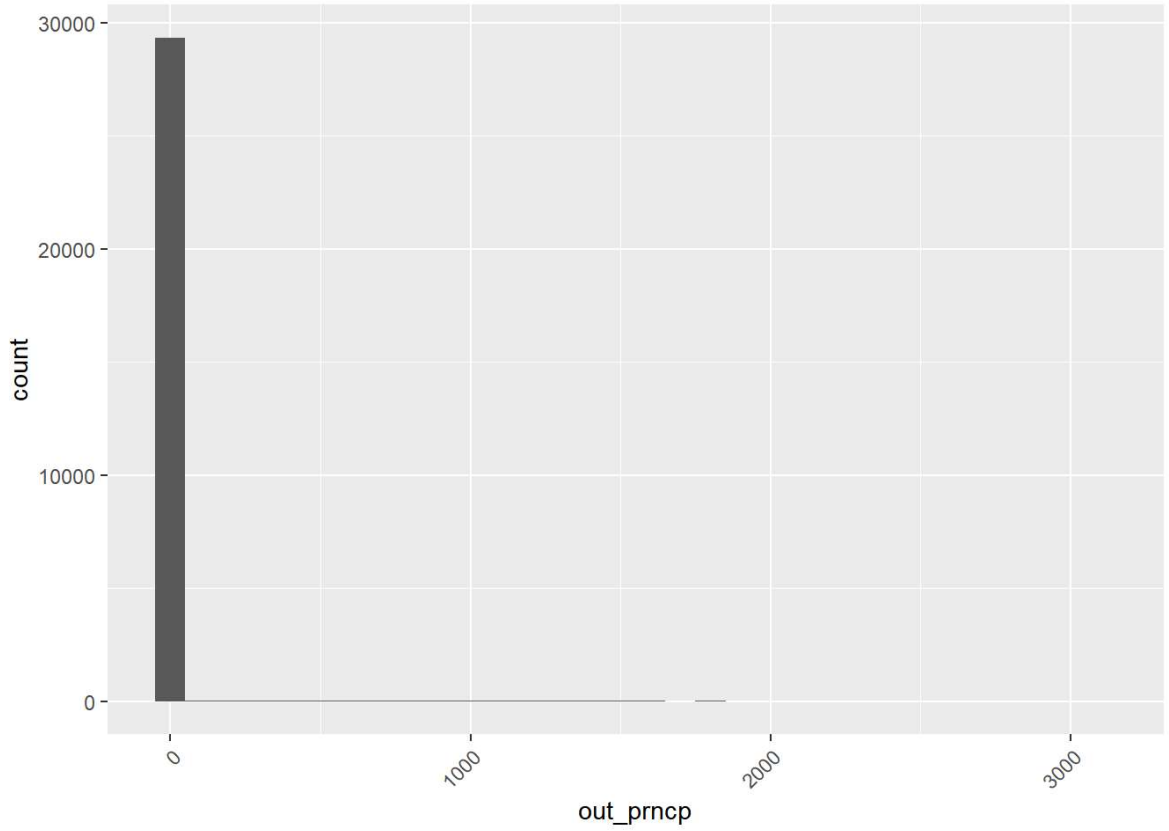

```
loan_without_outlier %>% ggplot(aes(x=revol_util)) + geom_histogram(binwidth=0.05) + theme(axis.text.x=element_text(angle=45,
        hjust=1))
```



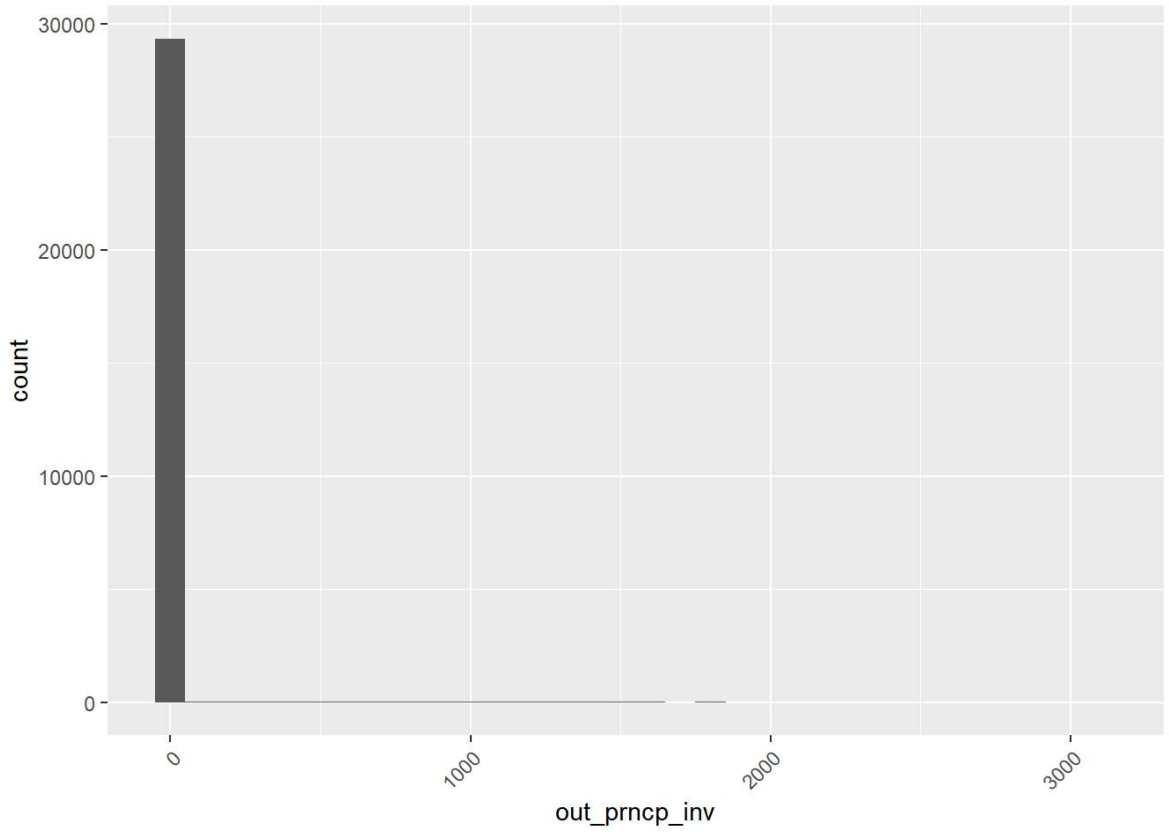

```
loan_without_outlier %>% ggplot(aes(x=total_acc)) + geom_histogram(binwidth=2) + theme(axis.text.x=element_text(angle=45, hju
          st=1))
```



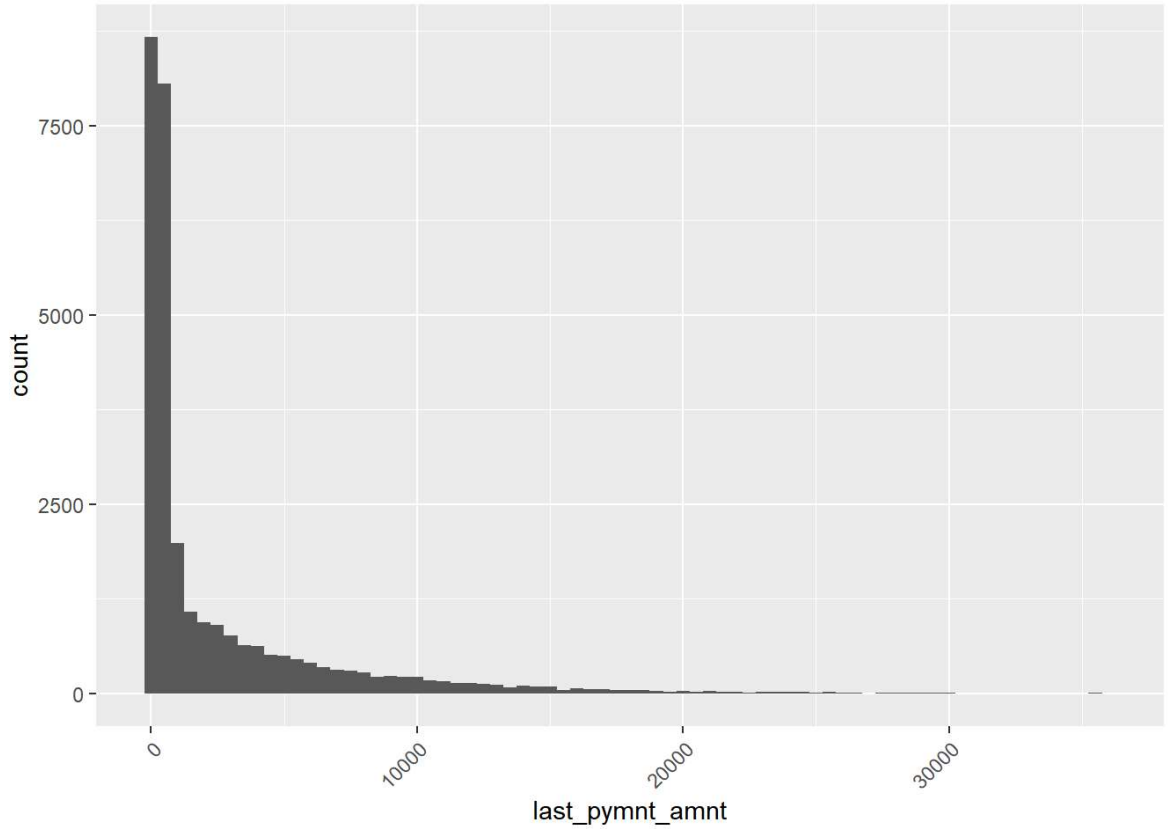

```
loan_without_outlier %>% ggplot(aes(x=out_prncp)) + geom_histogram(binwidth=100) + theme(axis.text.x=element_text(angle=45, h
          just=1))
```



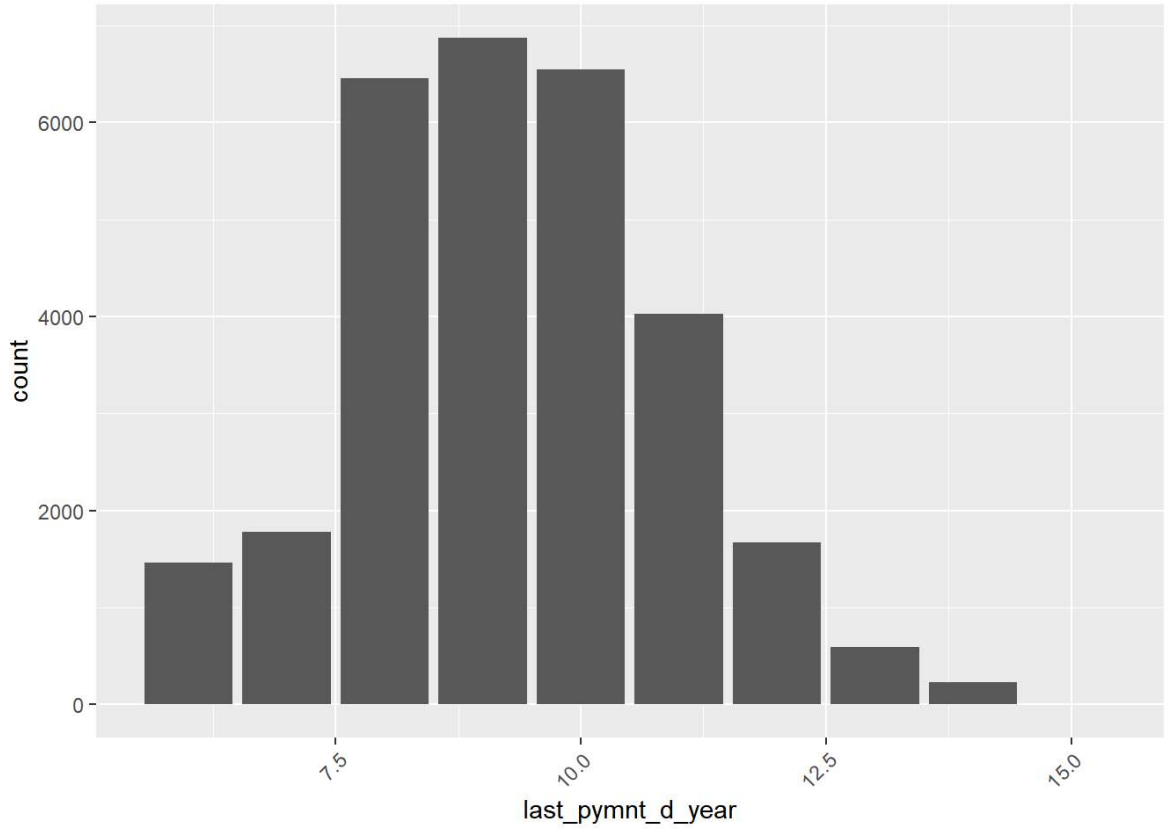

```
loan_without_outlier %>% ggplot(aes(x=out_prncp_inv)) + geom_histogram(binwidth=100) + theme(axis.text.x=element_text(angle=4
          5, hjust=1))
```



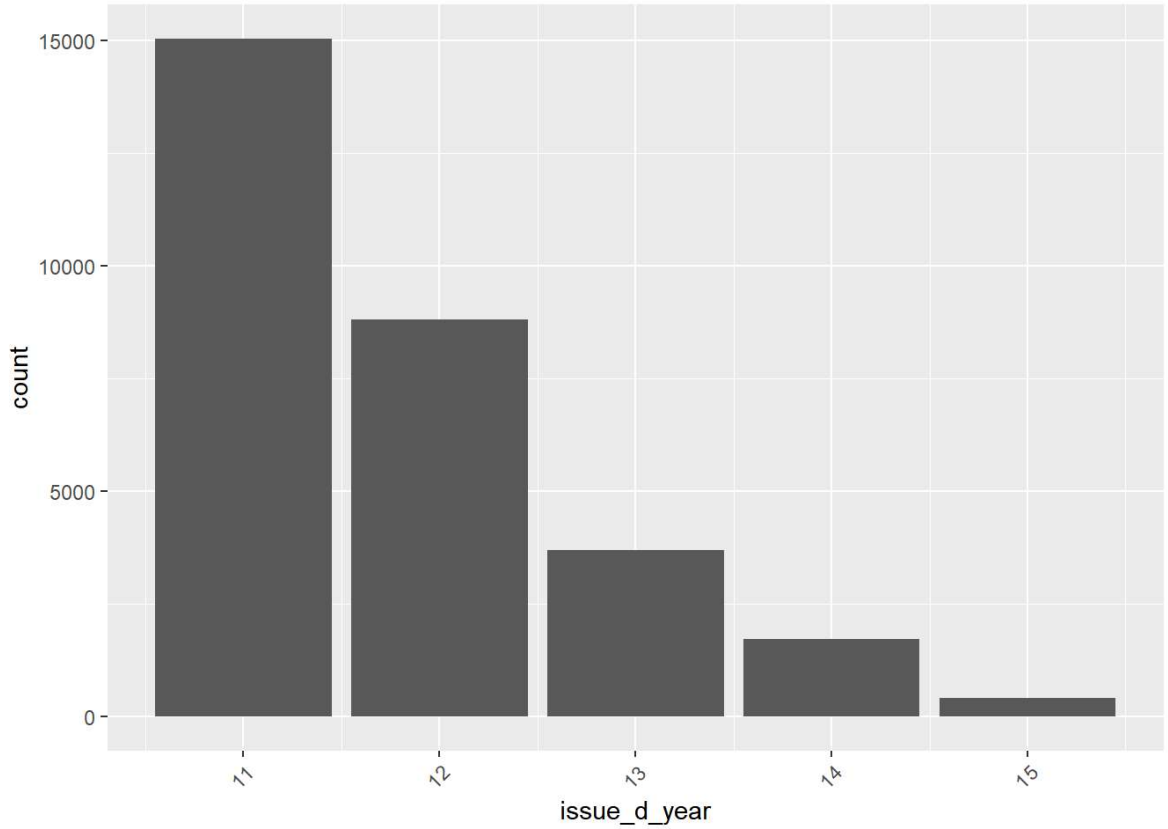

```
# remove variable: out_prncp, out_prncp_inv (only zero values)

loan_without_outlier %>% ggplot(aes(x=last_pymnt_amnt)) + geom_histogram(binwidth=500) + theme(axis.text.x=element_text(angle
          =45, hjust=1))
```



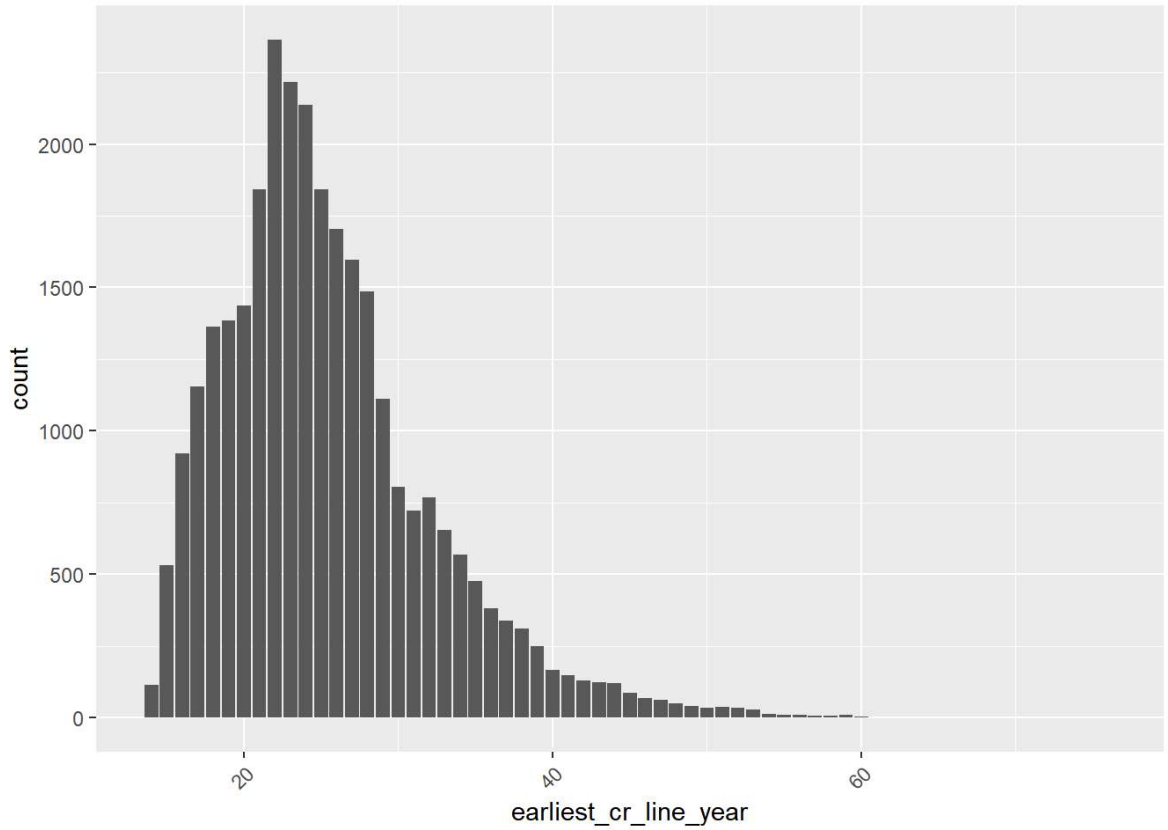

```
loan_without_outlier %>% ggplot(aes(x=last_pymnt_d_year)) + geom_bar() + theme(axis.text.x=element_text(angle=45, hjust=1))
```



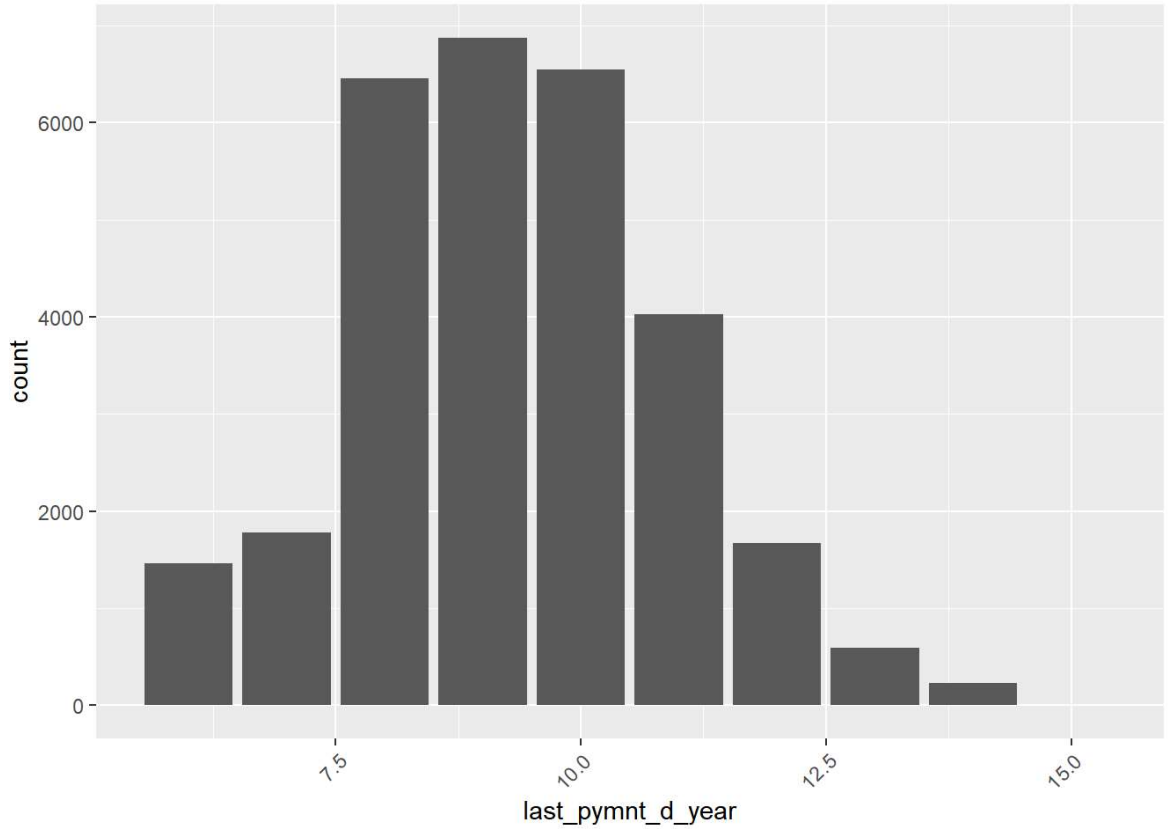

```
loan_without_outlier %>% ggplot(aes(x=issue_d_year)) + geom_bar() + theme(axis.text.x=element_text(angle=45, hjust=1))
```



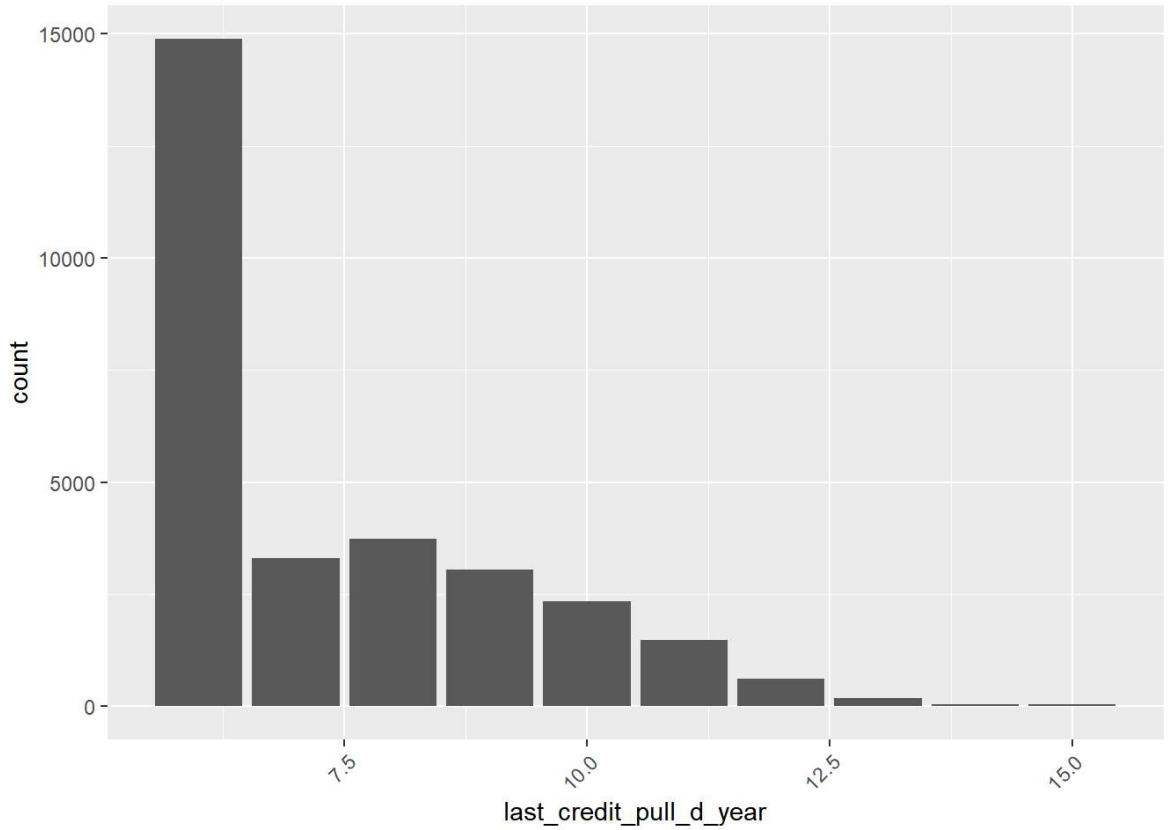

```
loan_without_outlier %>% ggplot(aes(x=earliest_cr_line_year)) + geom_bar() + theme(axis.text.x=element_text(angle=45, hjust=
        1))
```




```
loan_without_outlier %>% ggplot(aes(x=last_pymnt_d_year)) + geom_bar() + theme(axis.text.x=element_text(angle=45, hjust=1))
```



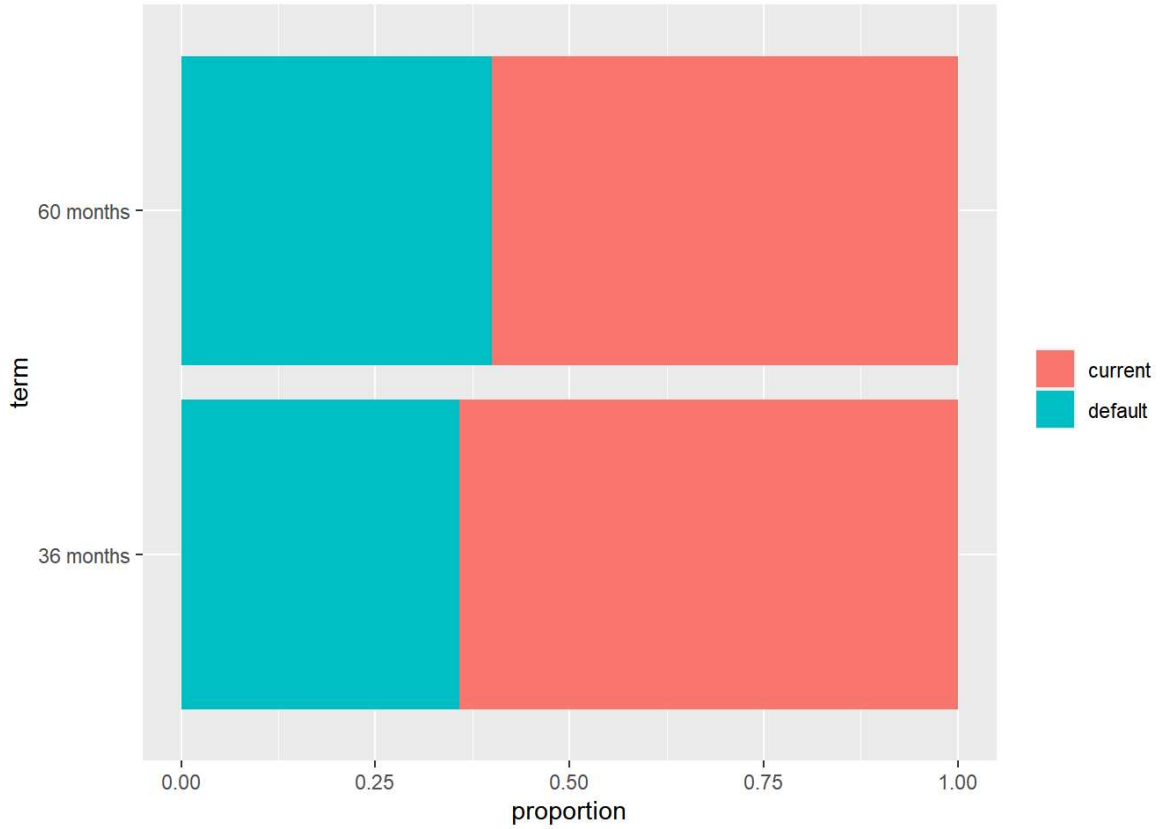

```
loan_without_outlier %>% ggplot(aes(x=last_credit_pull_d_year)) + geom_bar() + theme(axis.text.x=element_text(angle=45, hjust
=1))
```

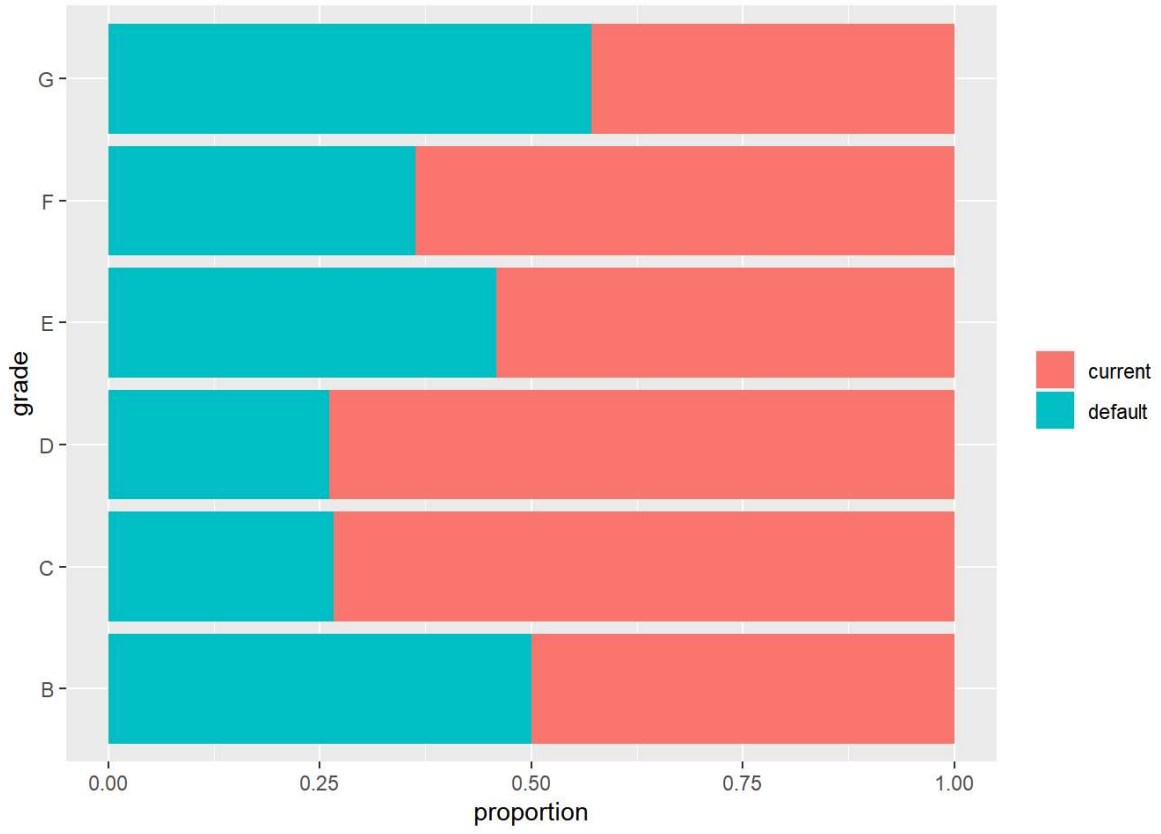


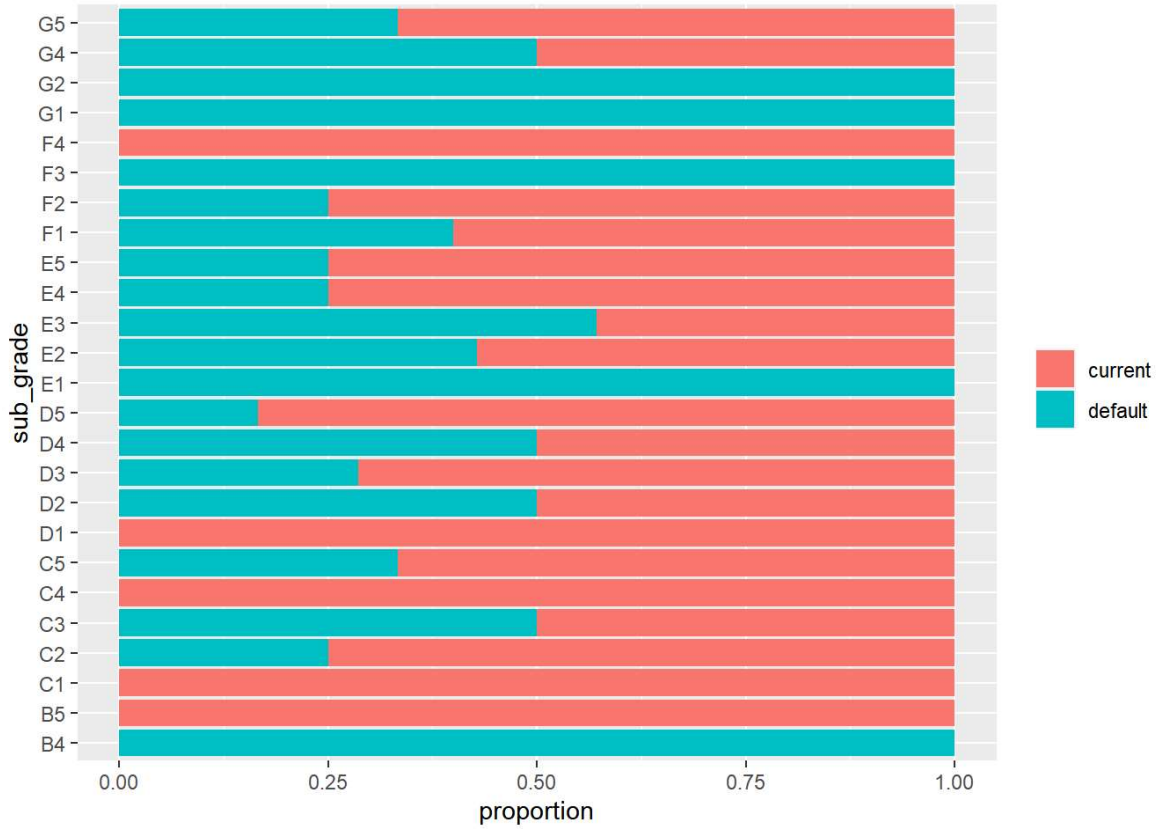
Explore Character Variables

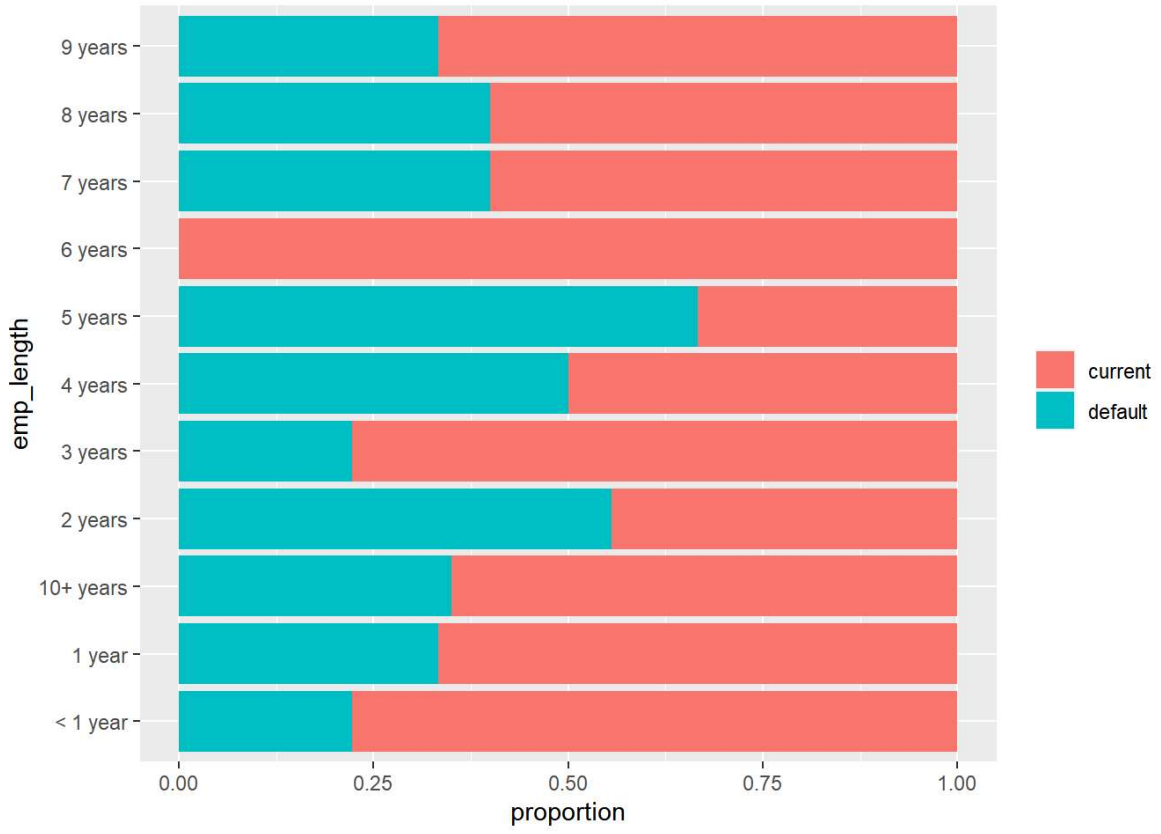
categorical variables: term, grade, sub_grade, emp_length, home_ownership, verification_status, pymnt_plan, purpose, addr_state, pub_rec, collections_12_mths_ex_med, policy_code, application_type, acc_now_delinq, chargeoff_within_12_mths, delinq_amnt, pub_rec_bankruptcies, tax_liens

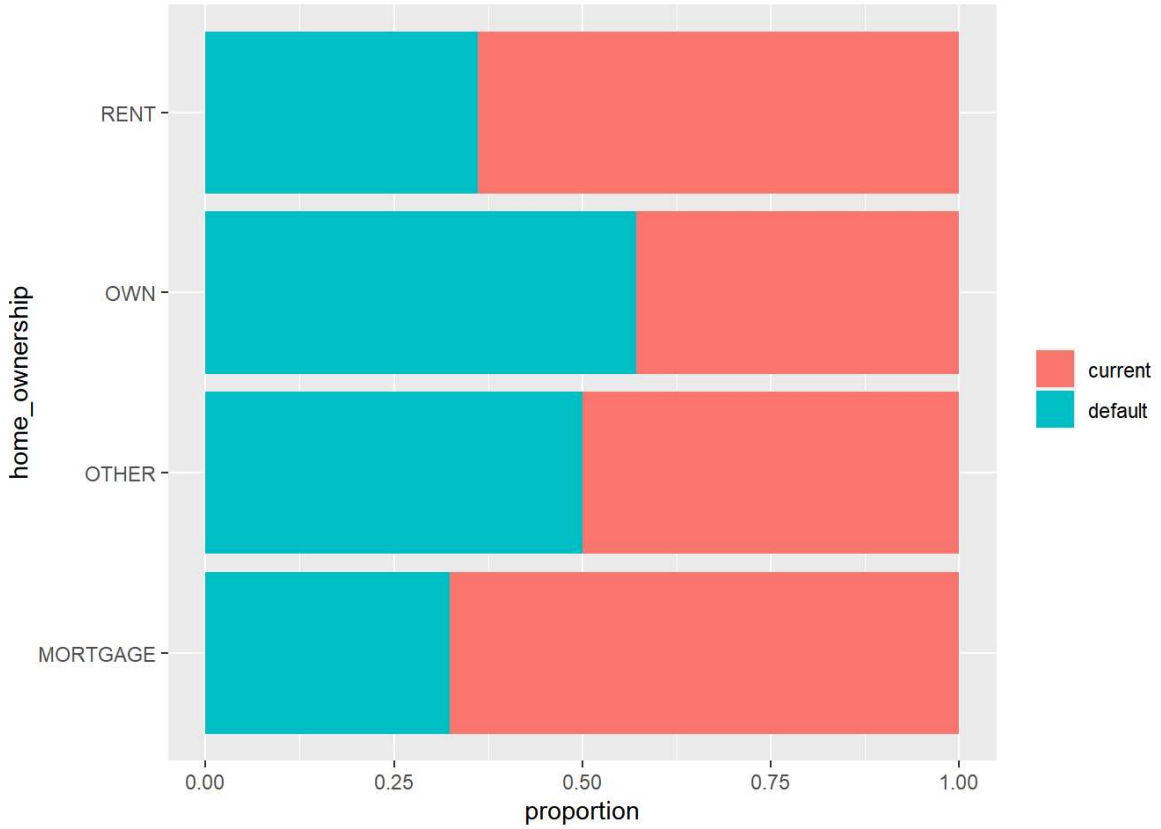
```
loan_decimal$pub_rec <- as.character(loan_decimal$pub_rec)
loan_decimal$collections_12_mths_ex_med <- as.character(loan_decimal$collections_12_mths_ex_med)
loan_decimal$policy_code <- as.character(loan_decimal$policy_code)
loan_decimal$application_type <- as.character(loan_decimal$application_type)
loan_decimal$acc_now_delinq <- as.character(loan_decimal$acc_now_delinq)
loan_decimal$chargeoff_within_12_mths <- as.character(loan_decimal$chargeoff_within_12_mths)
loan_decimal$delinq_amnt <- as.character(loan_decimal$delinq_amnt)
loan_decimal$tax_liens <- as.character(loan_decimal$tax_liens)


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


dummy <- c('term', 'grade', 'sub_grade', 'emp_length', 'home_ownership', 'verification_status', 'pymnt_plan', 'purpose', 'add
          r_state', 'pub_rec', 'collections_12_mths_ex_med', 'policy_code', 'application_type', 'acc_now_delinq', 'chargeoff_
          within_12_mths', 'delinq_amnt', 'pub_rec_bankruptcies', 'tax_liens')


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

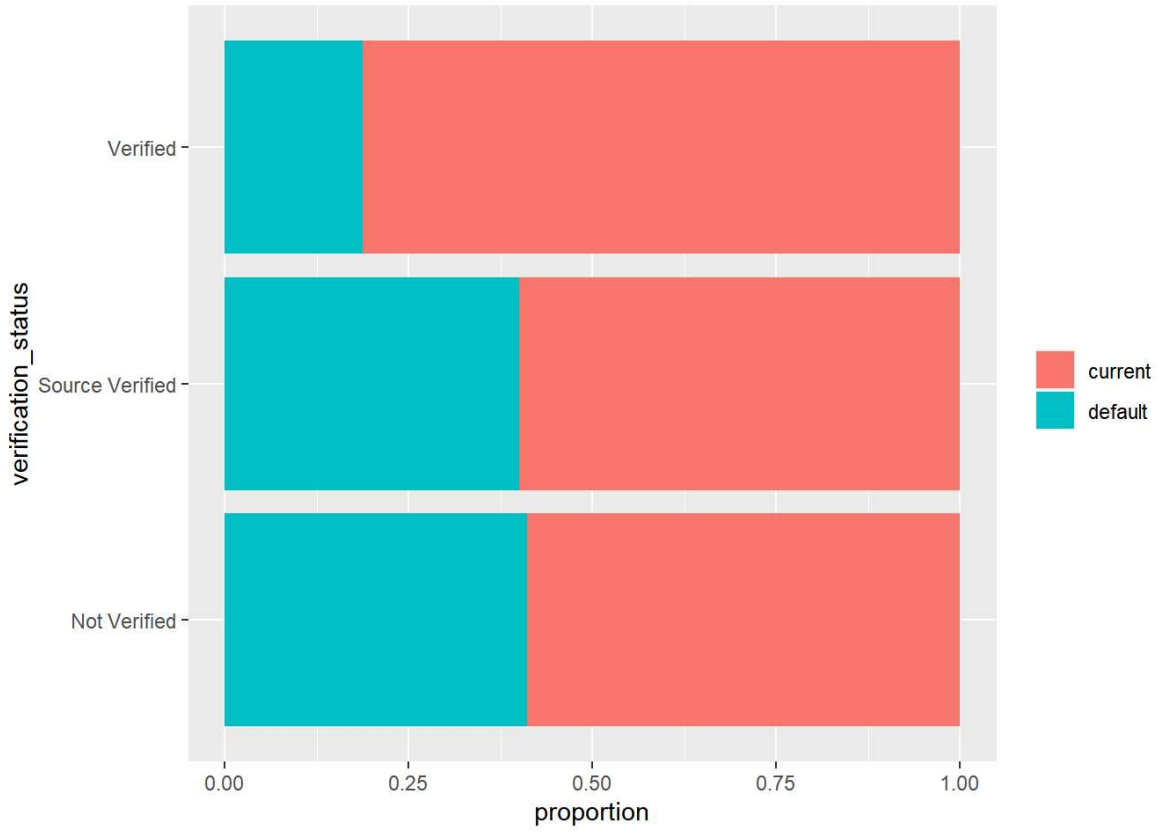

term
60 months
36 months
0.00
0.25
0.50
0.75
1.00
proportion
current
default

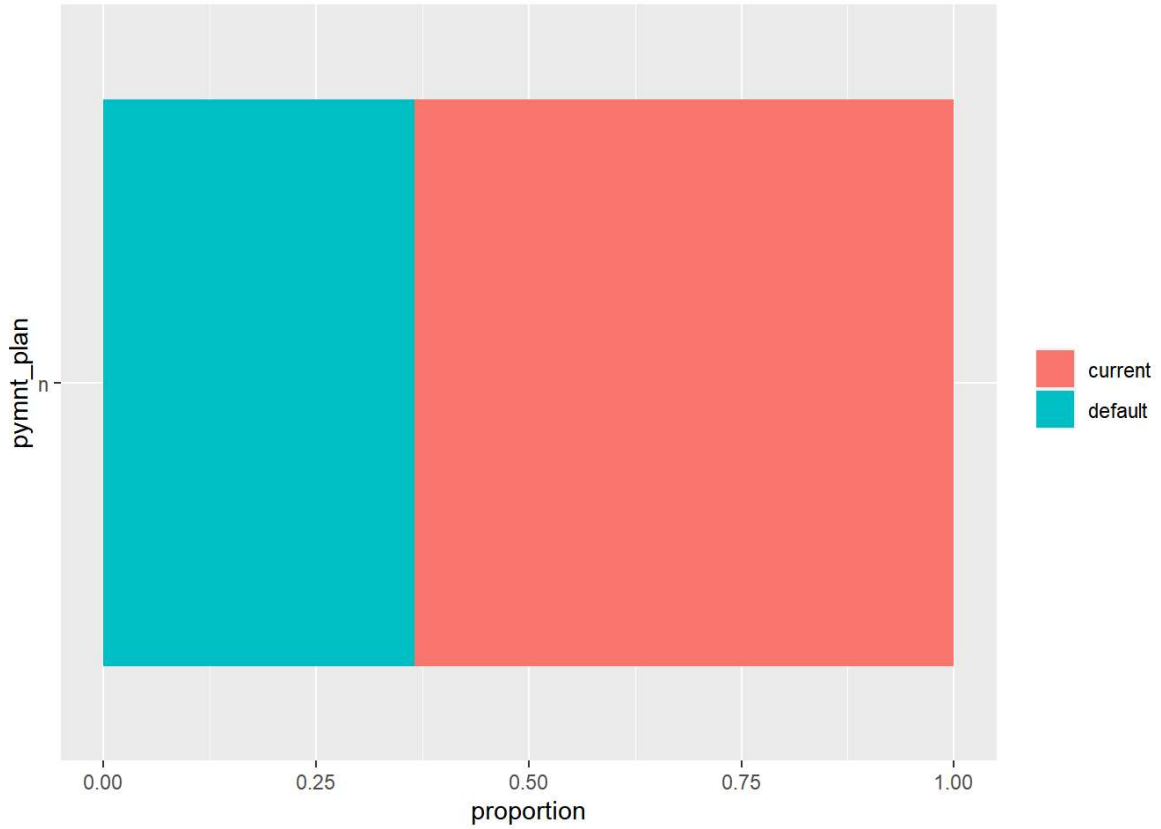

grade
G
F
E
D
C
B
0.00
0.25
0.50
0.75
1.00
proportion
current
default

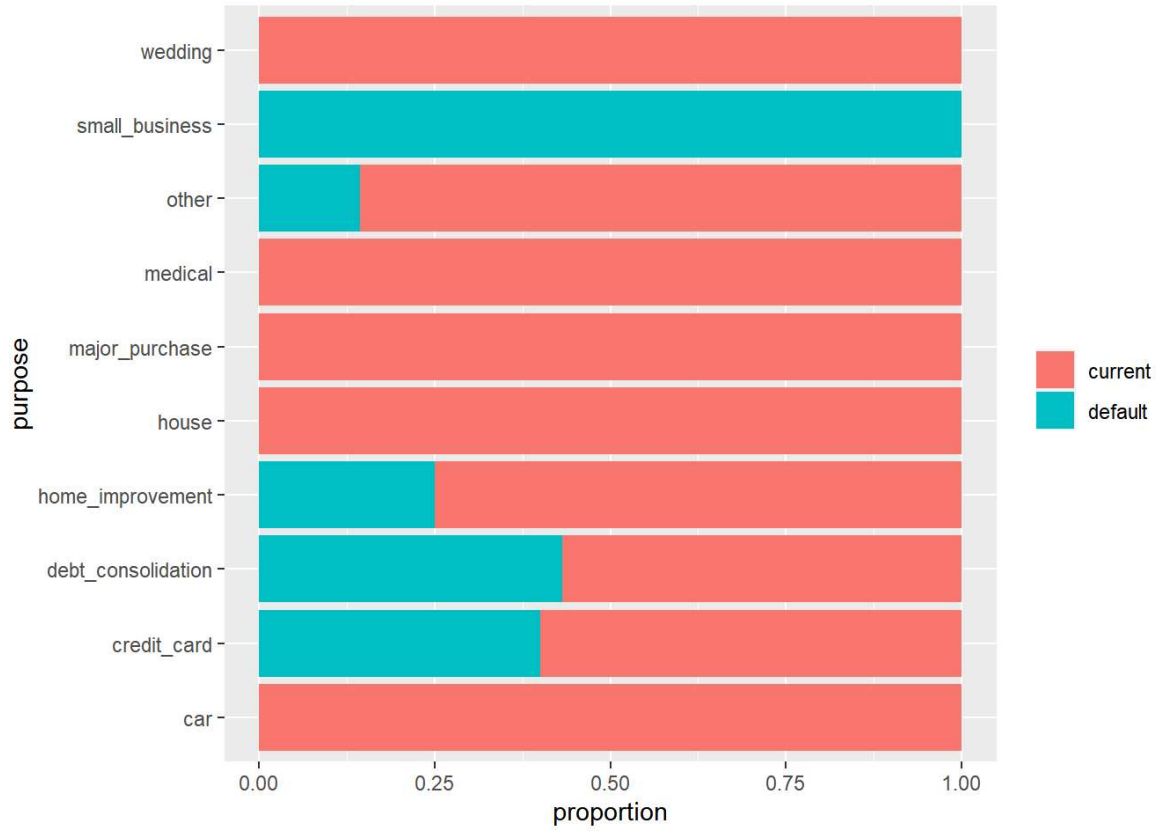

sub_grade
G5
G4
G2
G1
F4
F3
F2
F1
E5
E4
E3
E2
E1
D5
D4
D3
D2
D1
C5
C4
C3
C2
C1
B5
B4
0.00
0.25
0.50
0.75
1.00
proportion
current
default

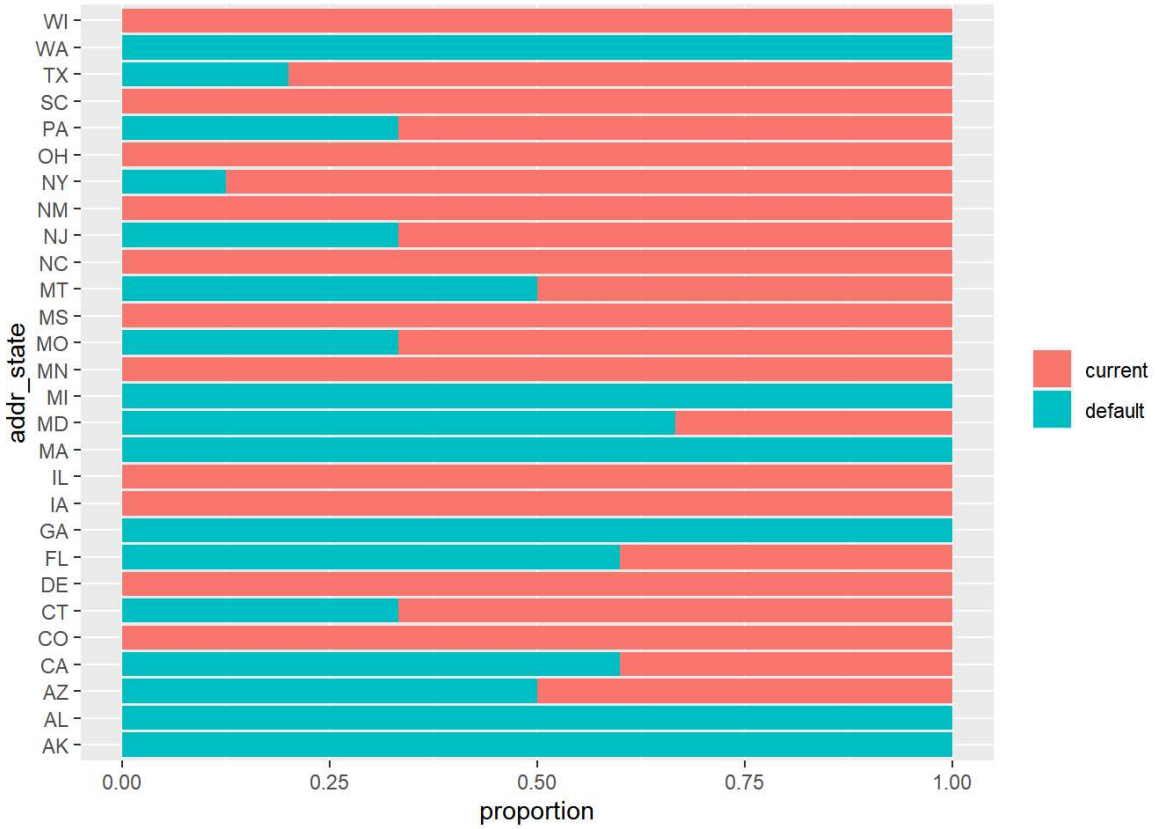

emp_length
9 years
8 years
7 years
6 years
5 years
4 years
3 years
2 years
10+ years
1 year
< 1 year
0.00
0.25
0.50
0.75
1.00
proportion
current
default

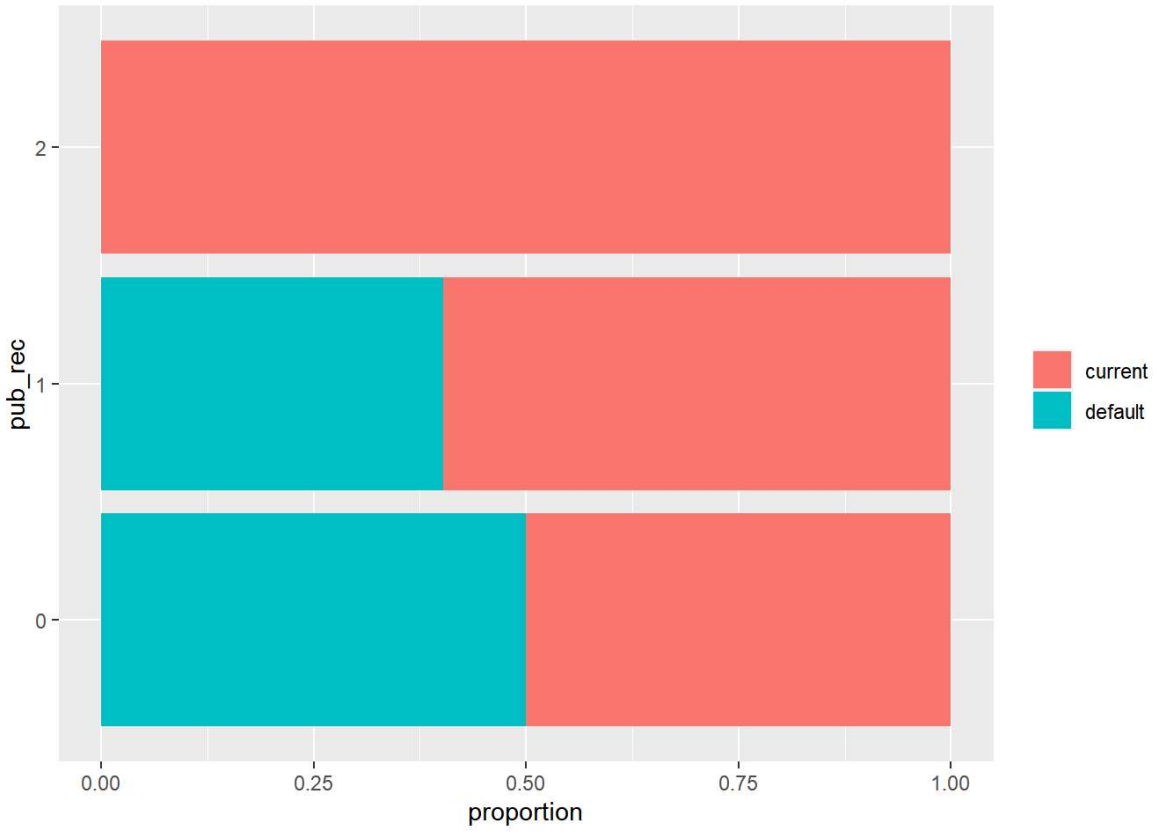

home_ownership
RENT
OWN
OTHER
MORTGAGE
0.00
0.25
0.50
0.75
1.00
proportion
current
default

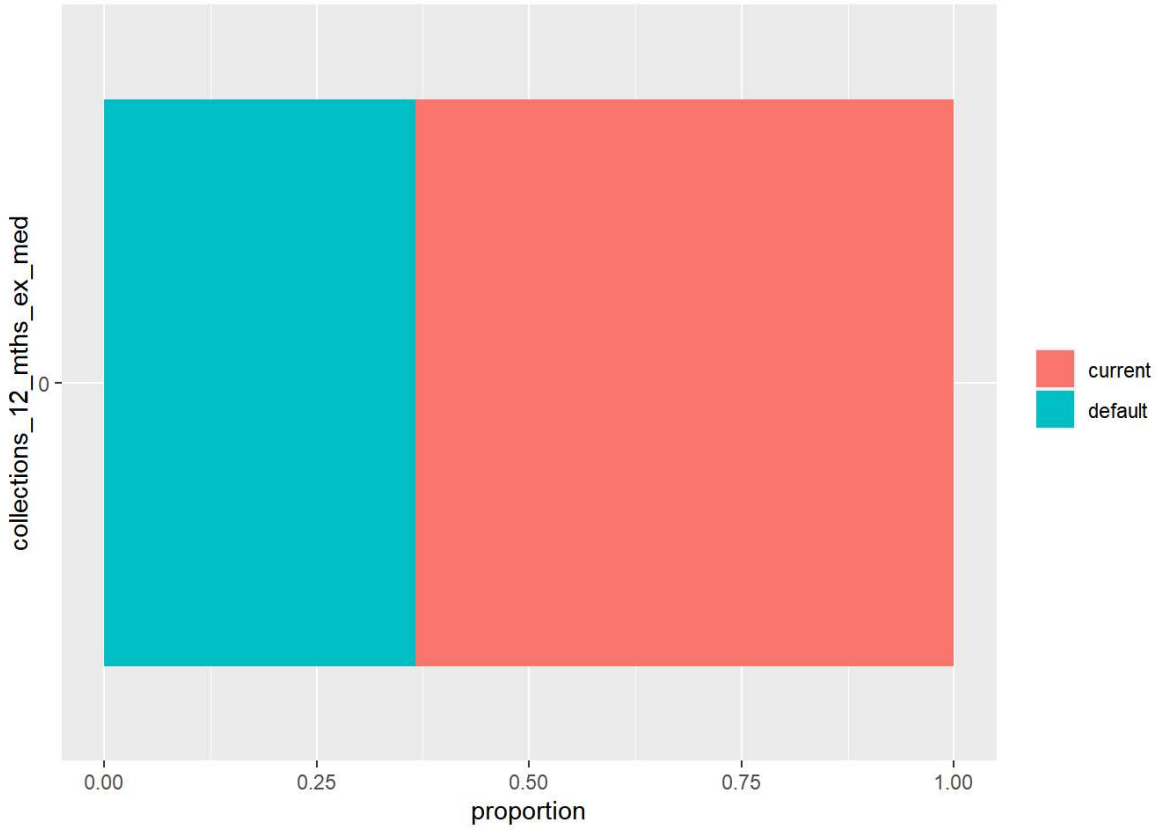

verification_status
Verified
Source Verified
Not Verified
0.00
0.25
0.50
0.75
1.00
proportion
current
default

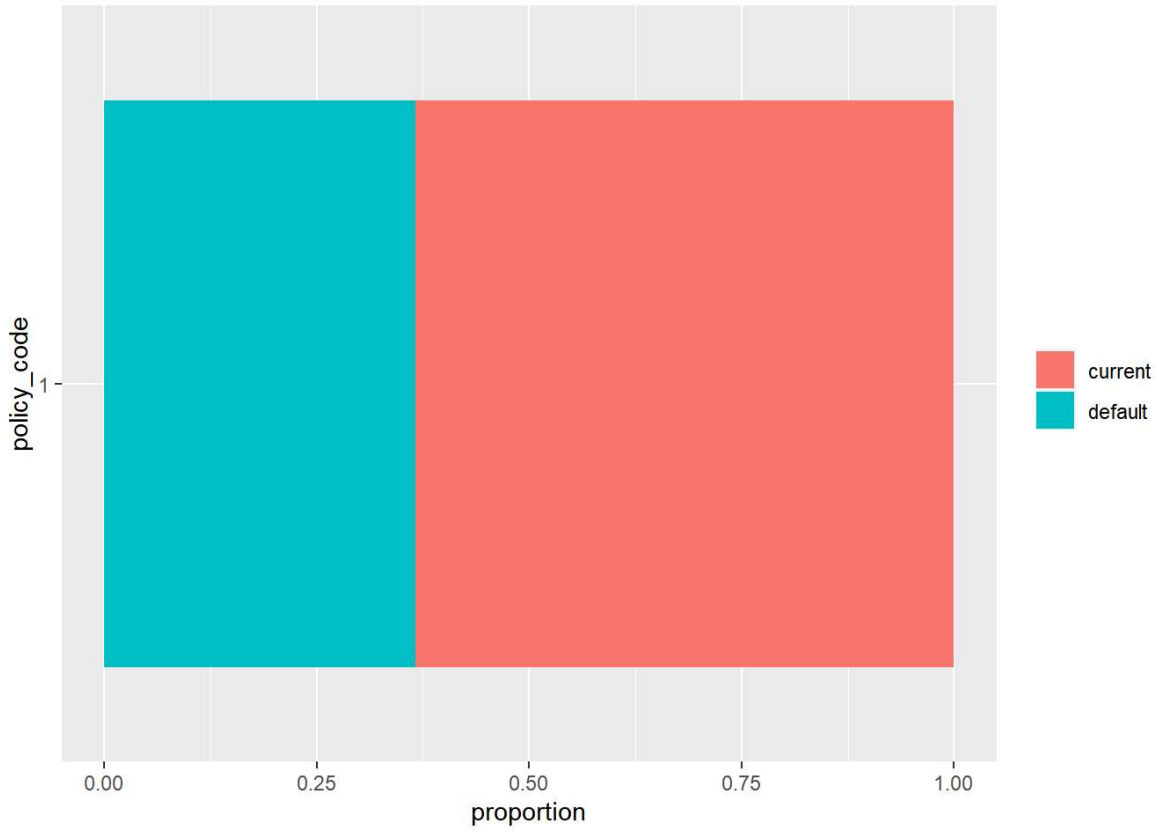

pymnt_plan
n
0.00
0.25
0.50
0.75
1.00
proportion
current
default

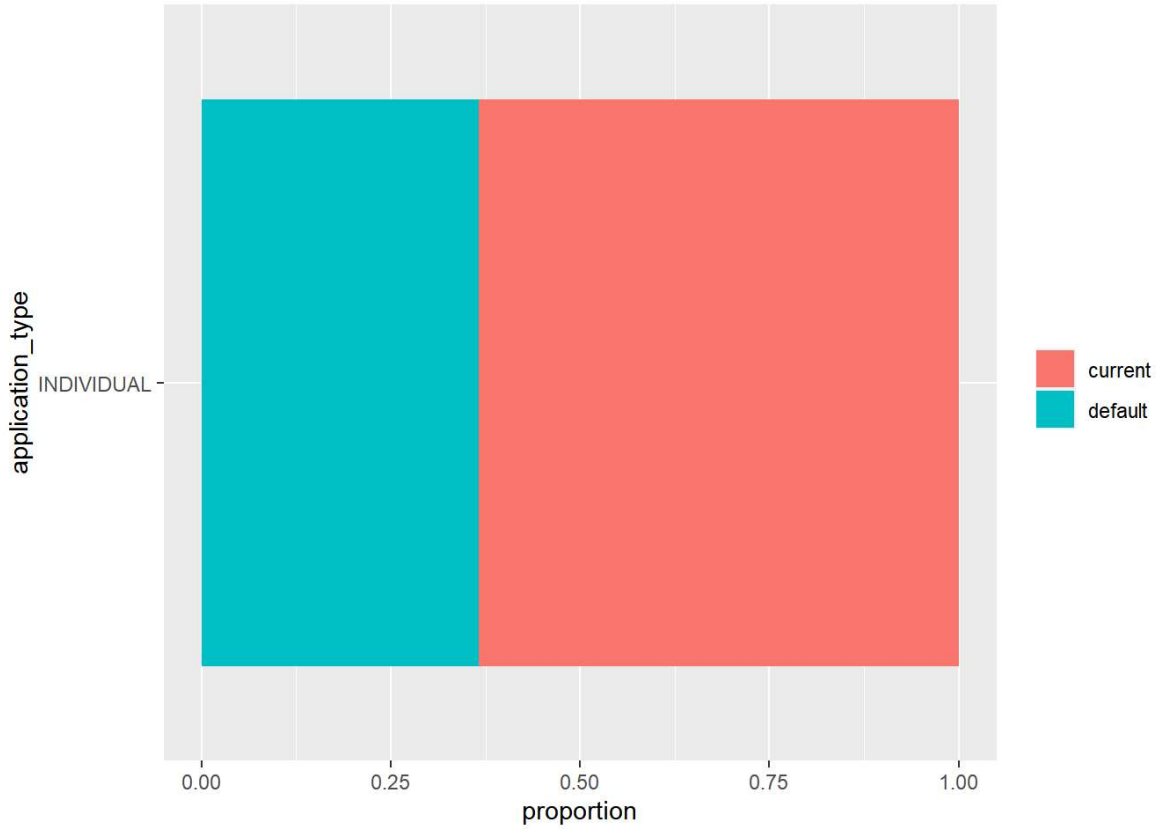

purpose
wedding
small_business
other
medical
major_purchase
house
home_improvement
debt_consolidation
credit_card
car
0.00
0.25
0.50
0.75
1.00
proportion
current
default

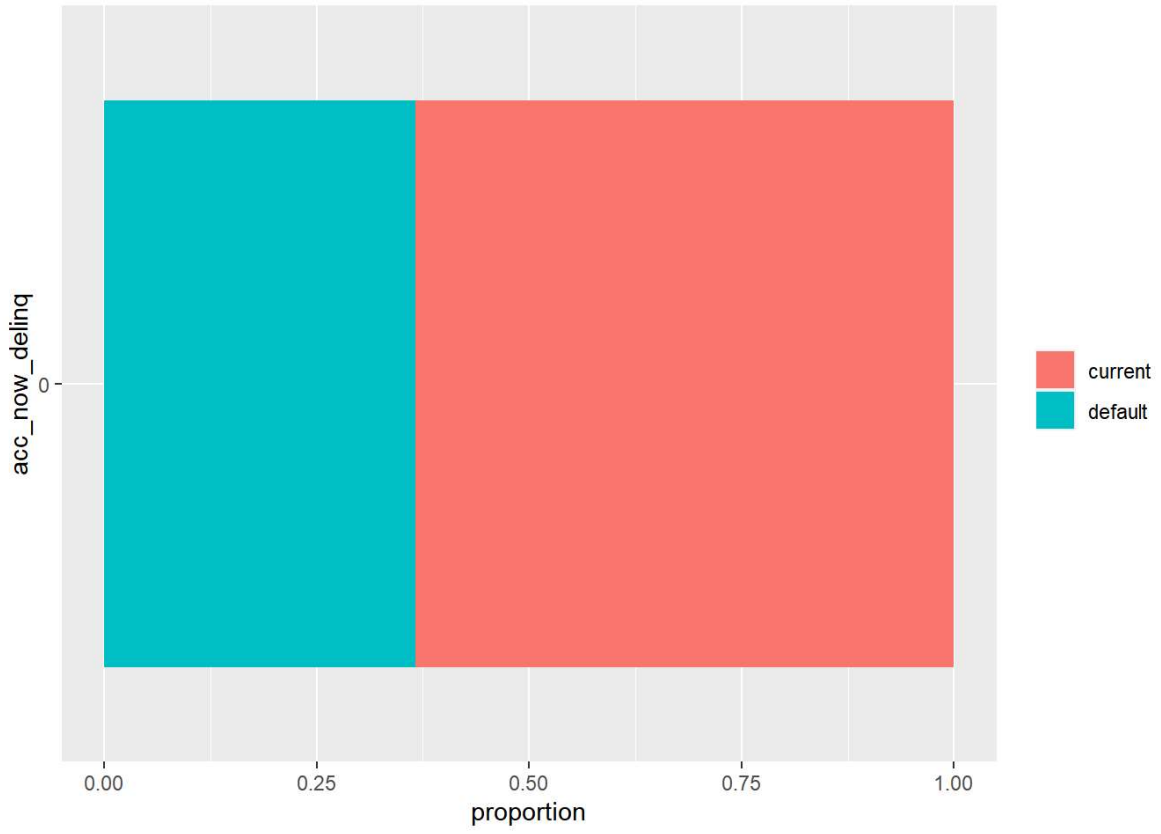

addr_state
WI
WA
TX
SC
PA
OH
NY
NM
NJ
NC
MT
MS
MO
MN
MI
MD
MA
IL
IA
GA
FL
DE
CT
CO
CA
AZ
AL
AK
0.00
0.25
0.50
0.75
1.00
proportion
current
default

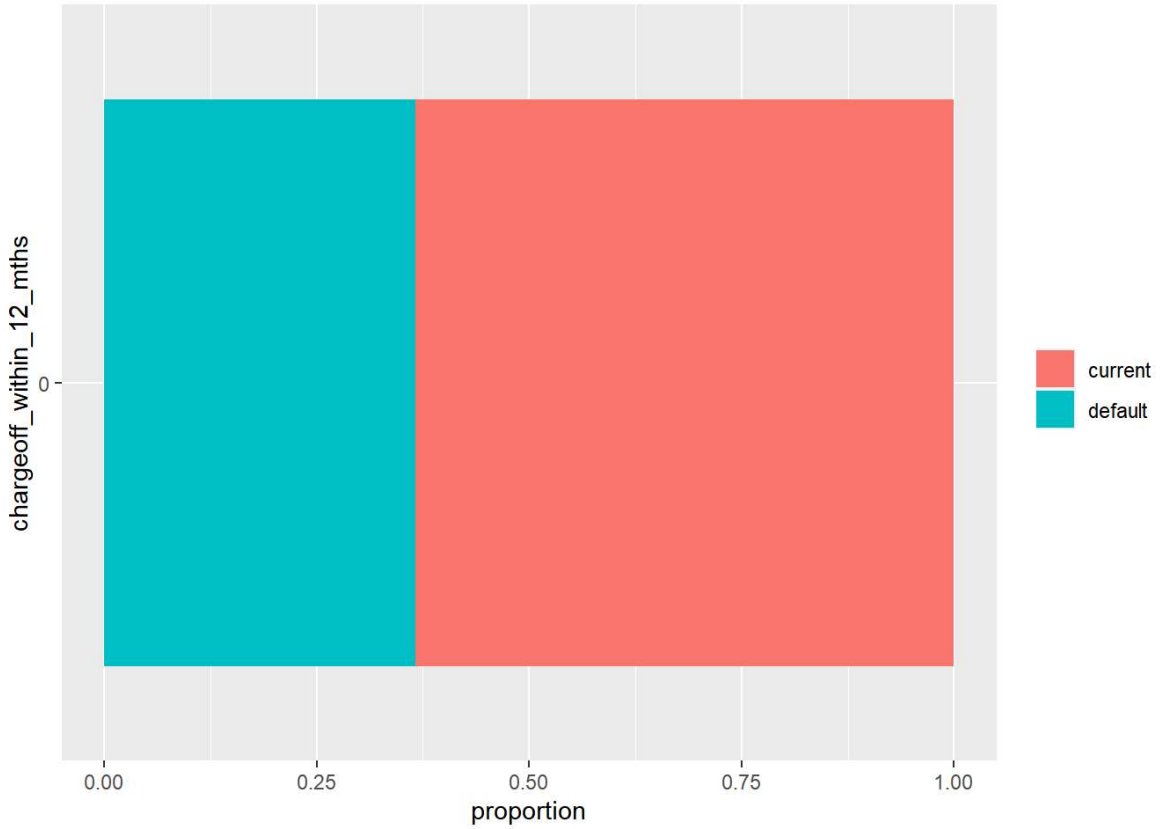

pub_rec
2
1
0
0.00
0.25
0.50
0.75
1.00
proportion
current
default

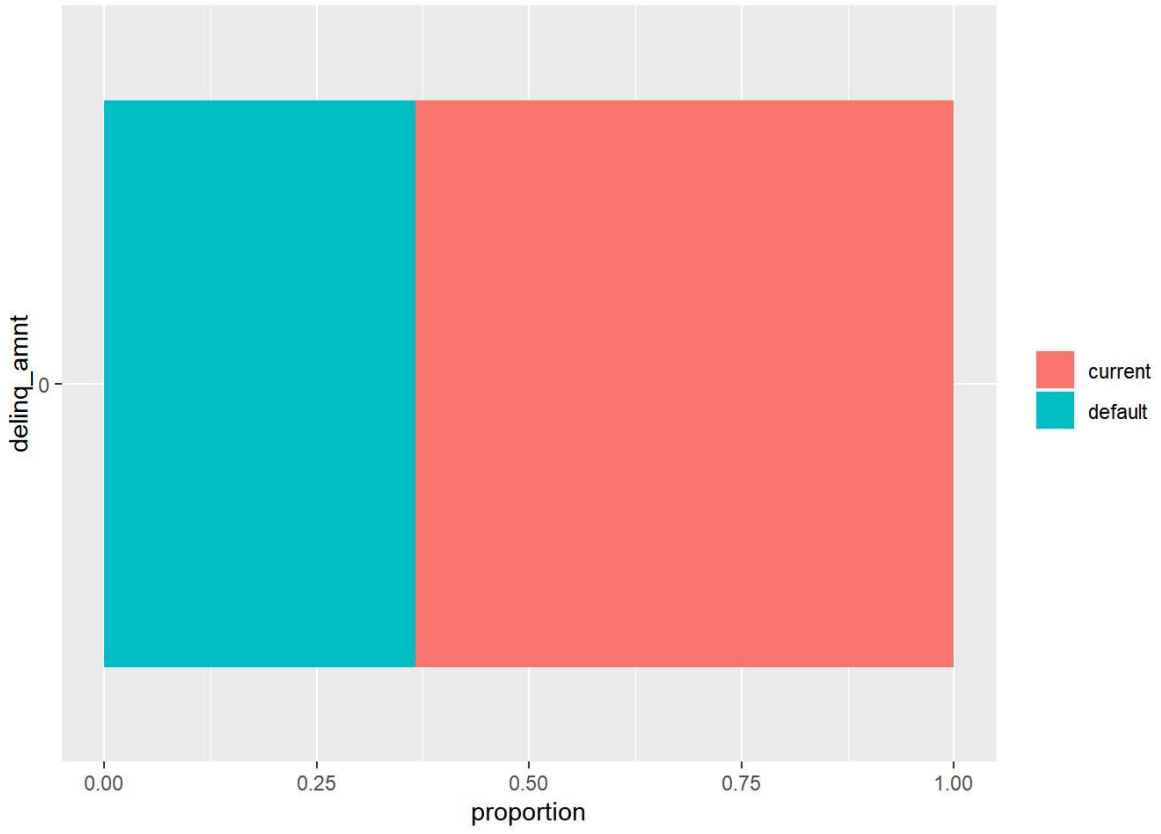

collections_12_mths_ex_med
0
0.00
0.25
0.50
0.75
1.00
proportion
current
default

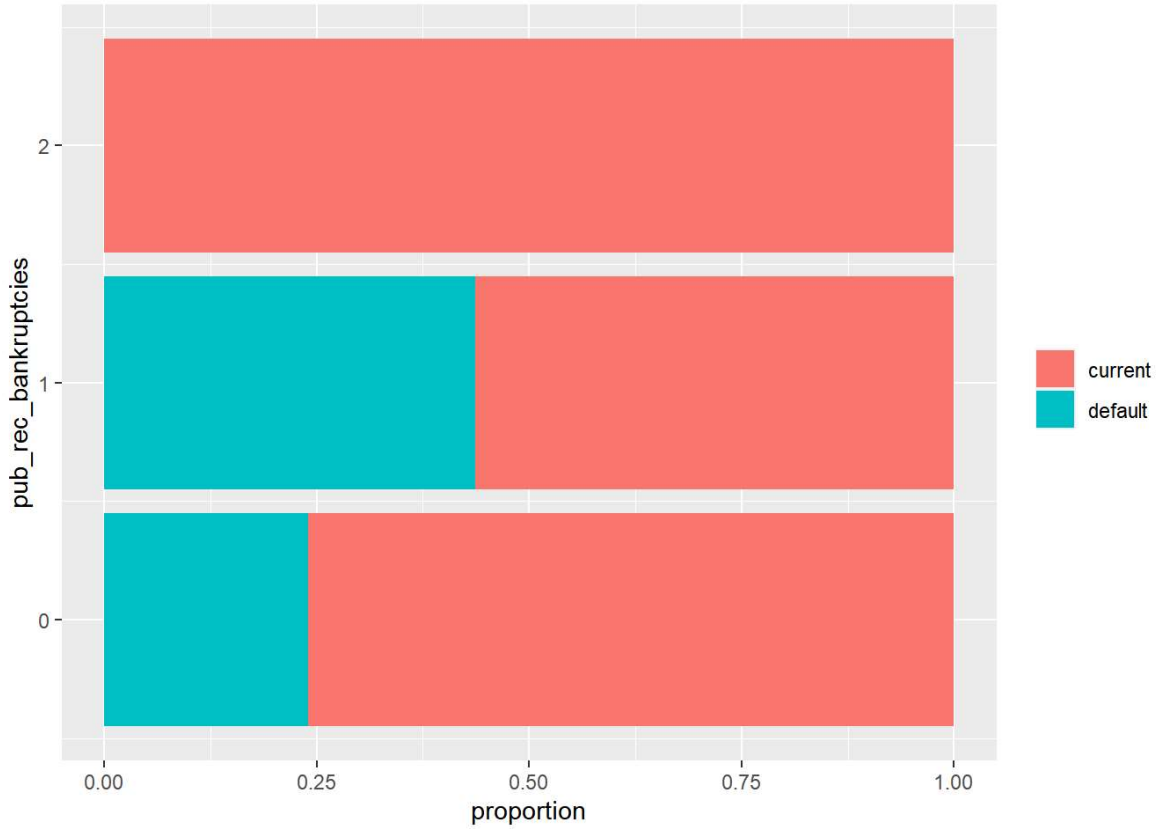

policy_code
1
0.00
0.25
0.50
0.75
1.00
proportion
current
default

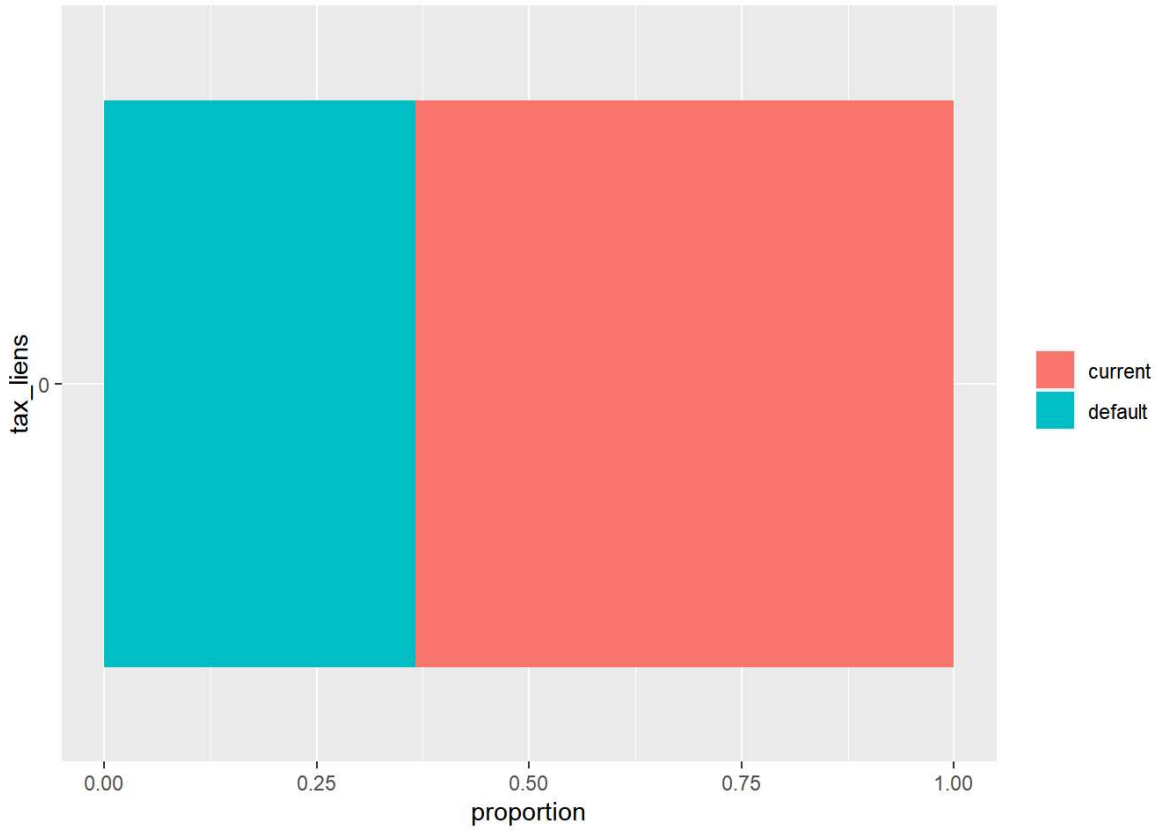

application_type
INDIVIDUAL
0.00
0.25
0.50
0.75
1.00
proportion
current
default


acc_now_delinq
0
0.00
0.25
0.50
0.75
1.00
proportion
current
default

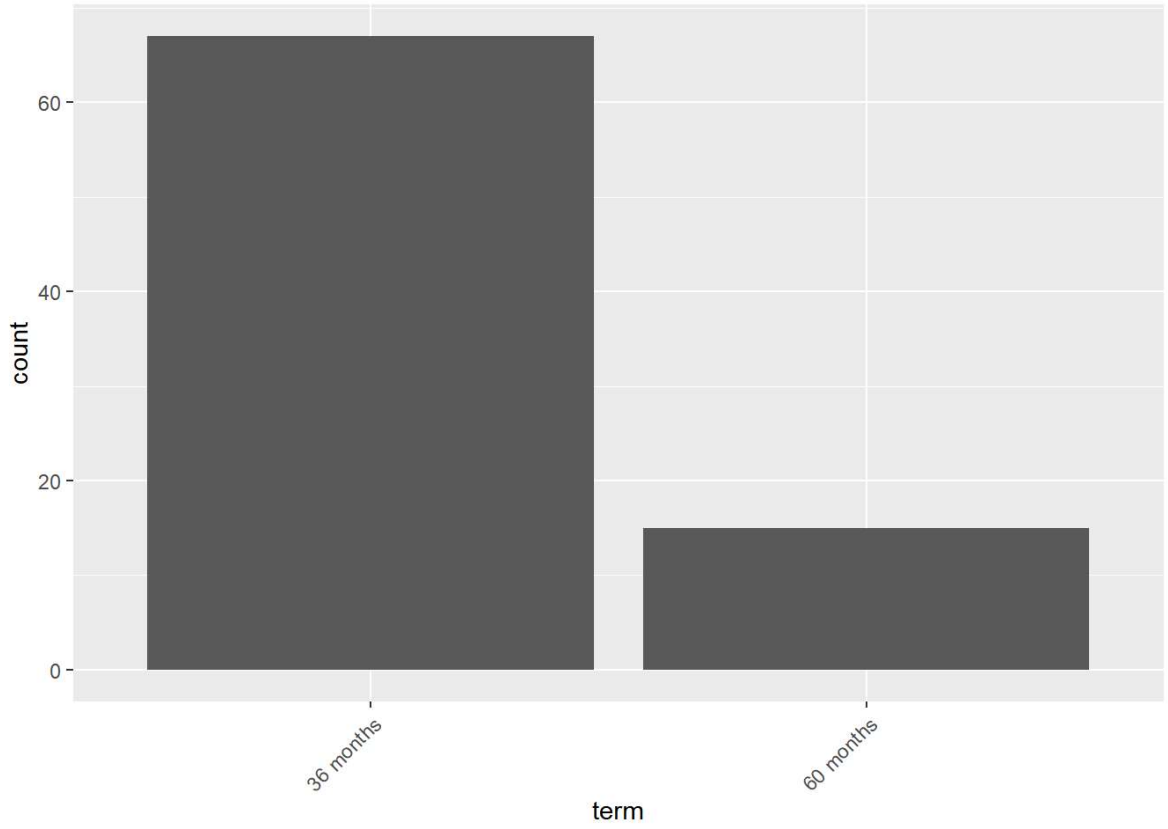

chargeoff_within_12_mths
0
0.00
0.25
0.50
0.75
1.00
proportion
current
default

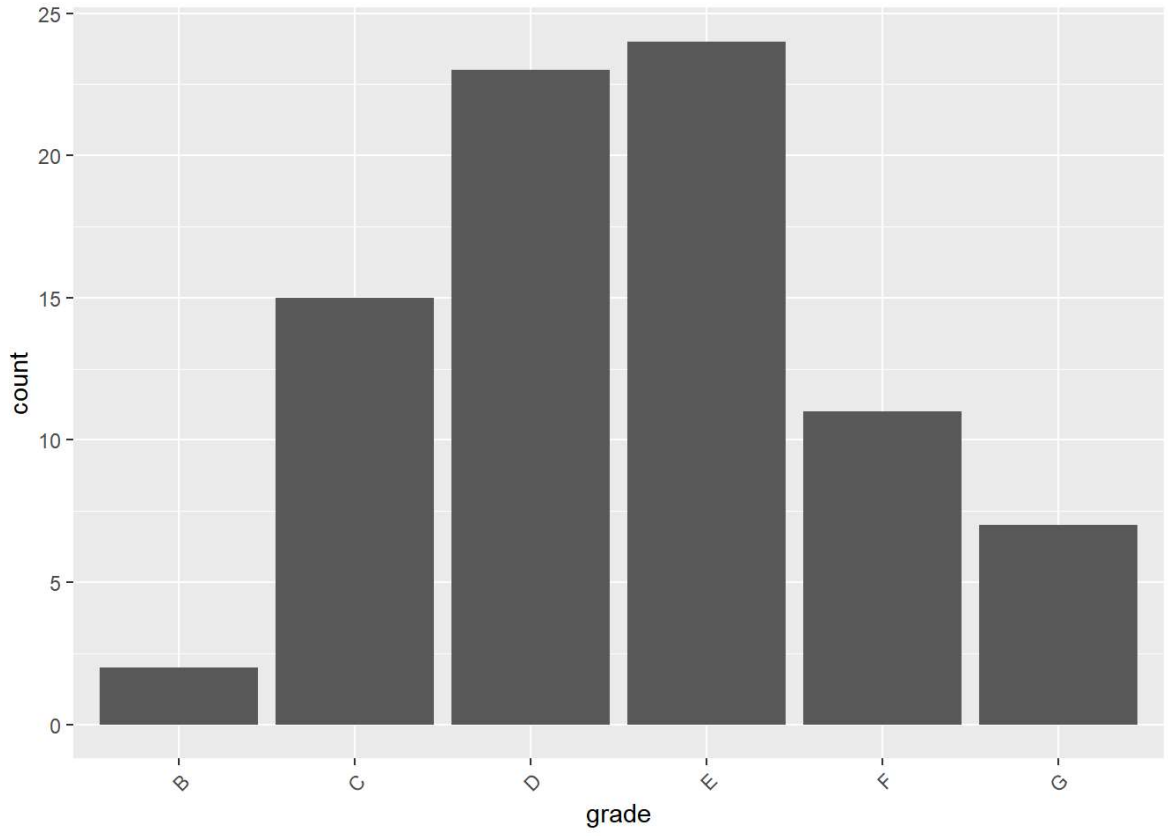

delinq_amnt
0
0.00
0.25
0.50
0.75
1.00
proportion
current
default

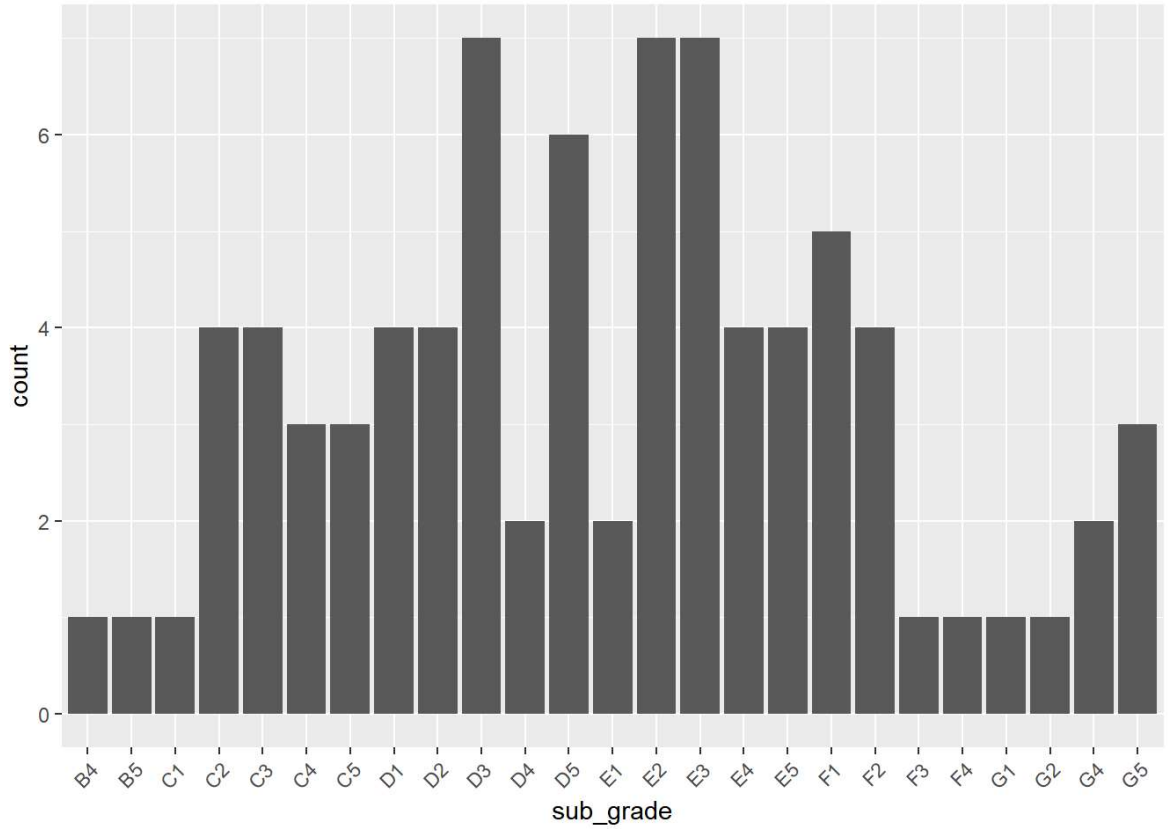

pub_rec_bankruptcies
2
1
0
0.00
0.25
0.50
0.75
1.00
proportion
current
default

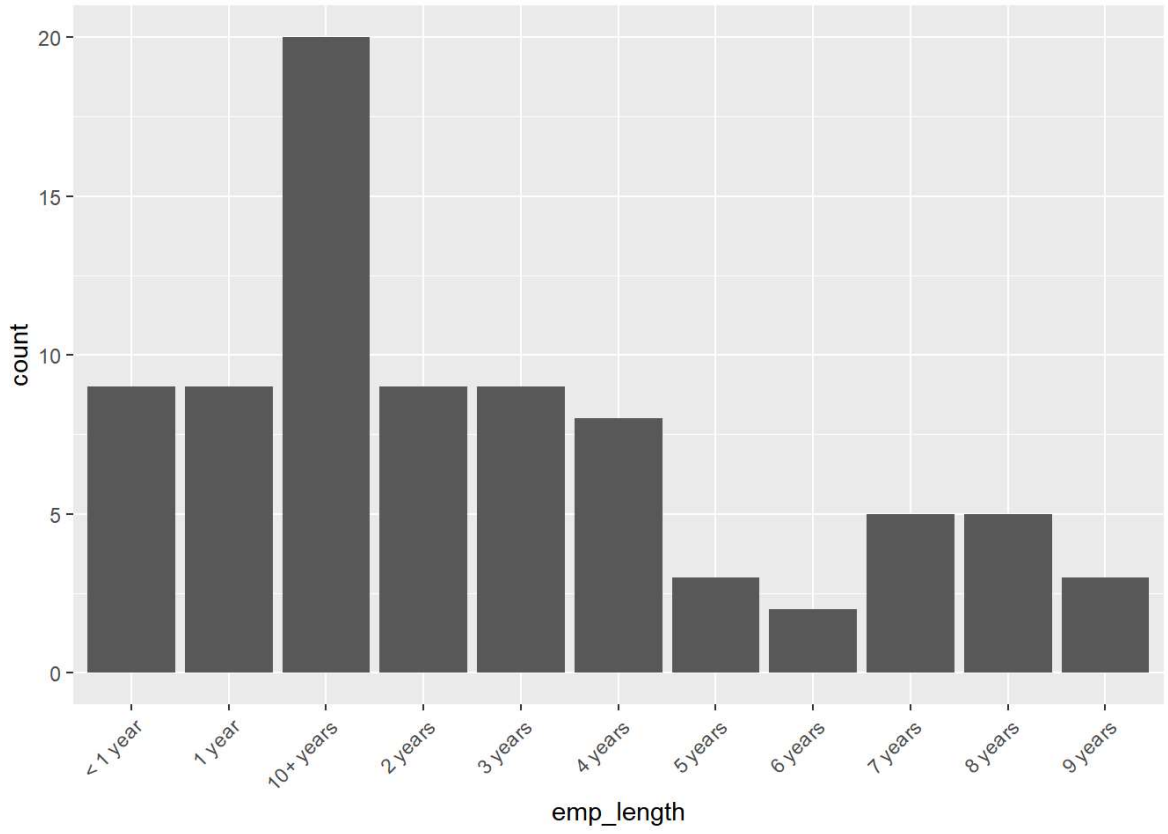

tax_liens
0
0.00
0.25
0.50
0.75
1.00
proportion
current
default

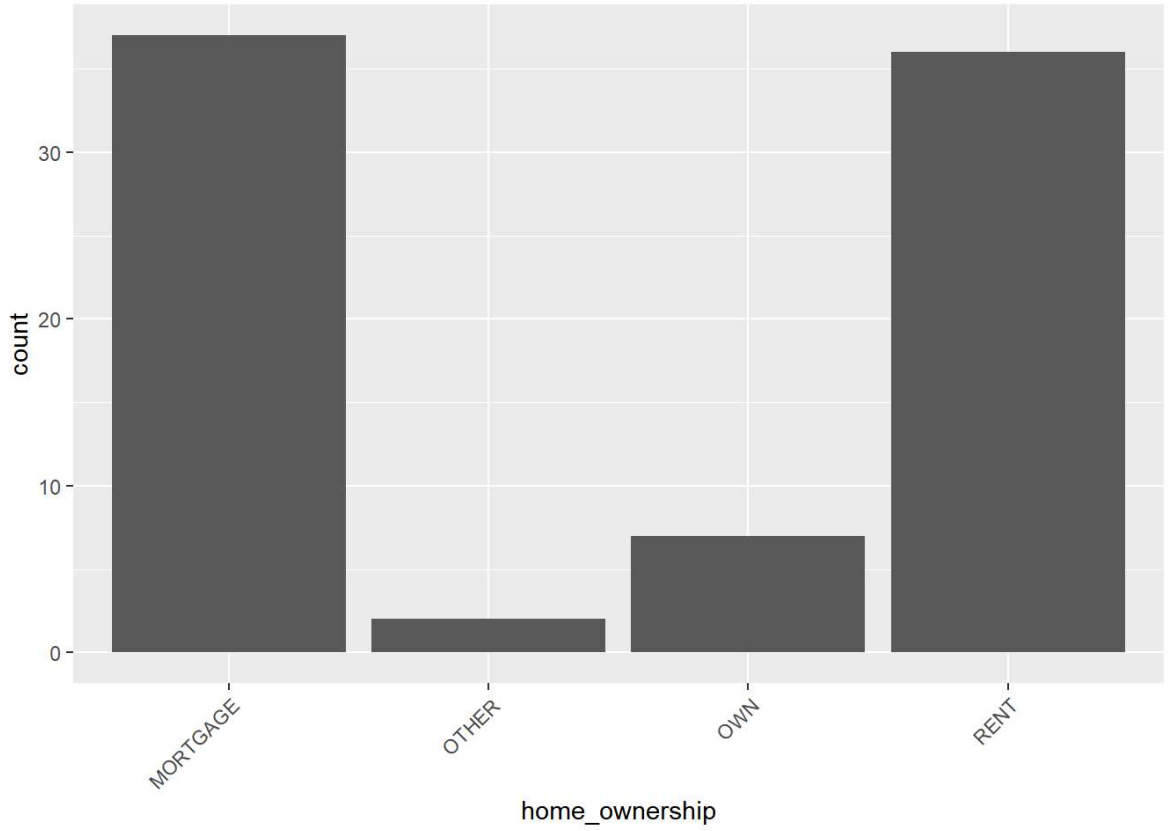
```
# drop variables with only one category / imbalanced variables: pymnt_plan, collections_12_mths_ex_med, policy_code, applicat
          ion_type, acc_now_delinq, chargeoff_within_12_mths, delinq_amnt, tax_liens


bar <- function(col){
    loan_decimal %>%
    na.omit() %>%
#    ggplot(aes(!!as.name(col))) +
#    geom_bar() +
    ggplot(aes(!!as.name(col)), fill = as.factor(loan_status)) +
    geom_bar(position = 'identity') +
    theme(axis.text.x=element_text(angle=45, hjust=1))
}


dummy <- c('term', 'grade', 'sub_grade', 'emp_length', 'home_ownership', 'verification_status', 'pymnt_plan', 'purpose', 'add
          r_state', 'pub_rec', 'collections_12_mths_ex_med', 'policy_code', 'application_type', 'acc_now_delinq', 'chargeoff_
          within_12_mths', 'delinq_amnt', 'pub_rec_bankruptcies', 'tax_liens')


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

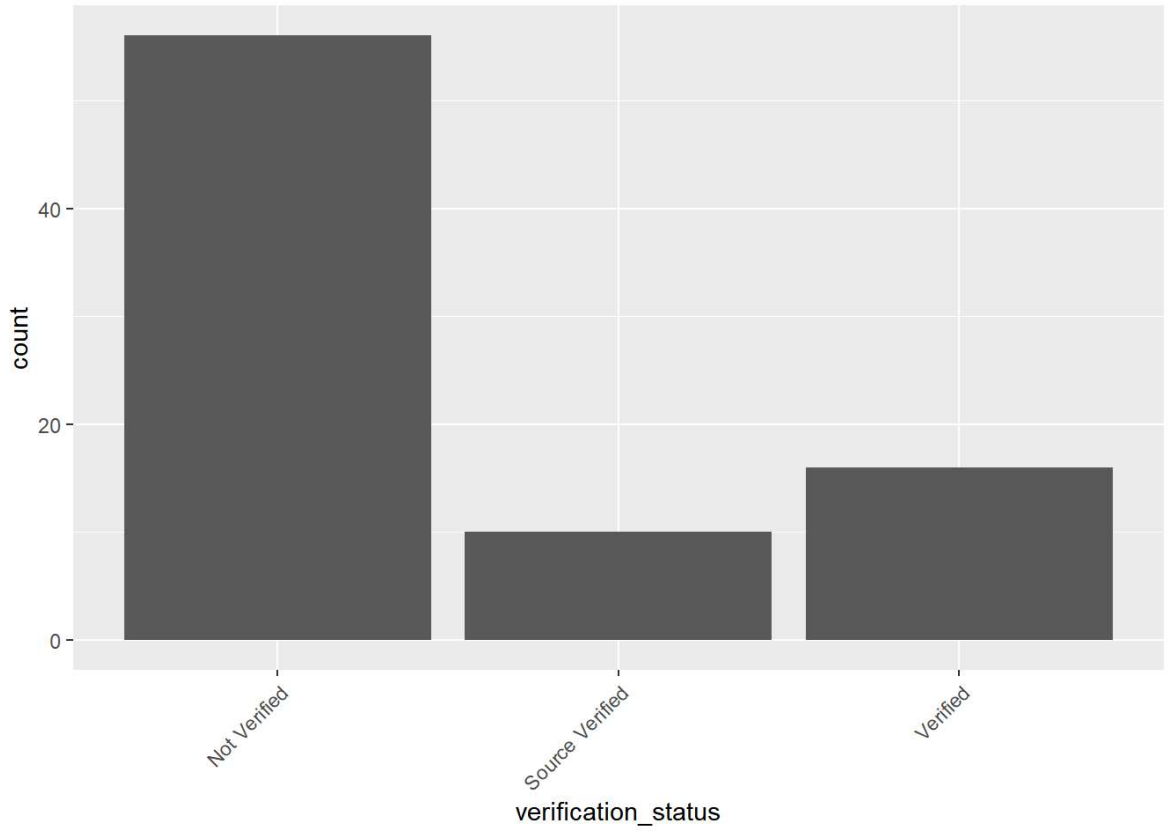

60
40
20
0
count
36 months
60 months
term

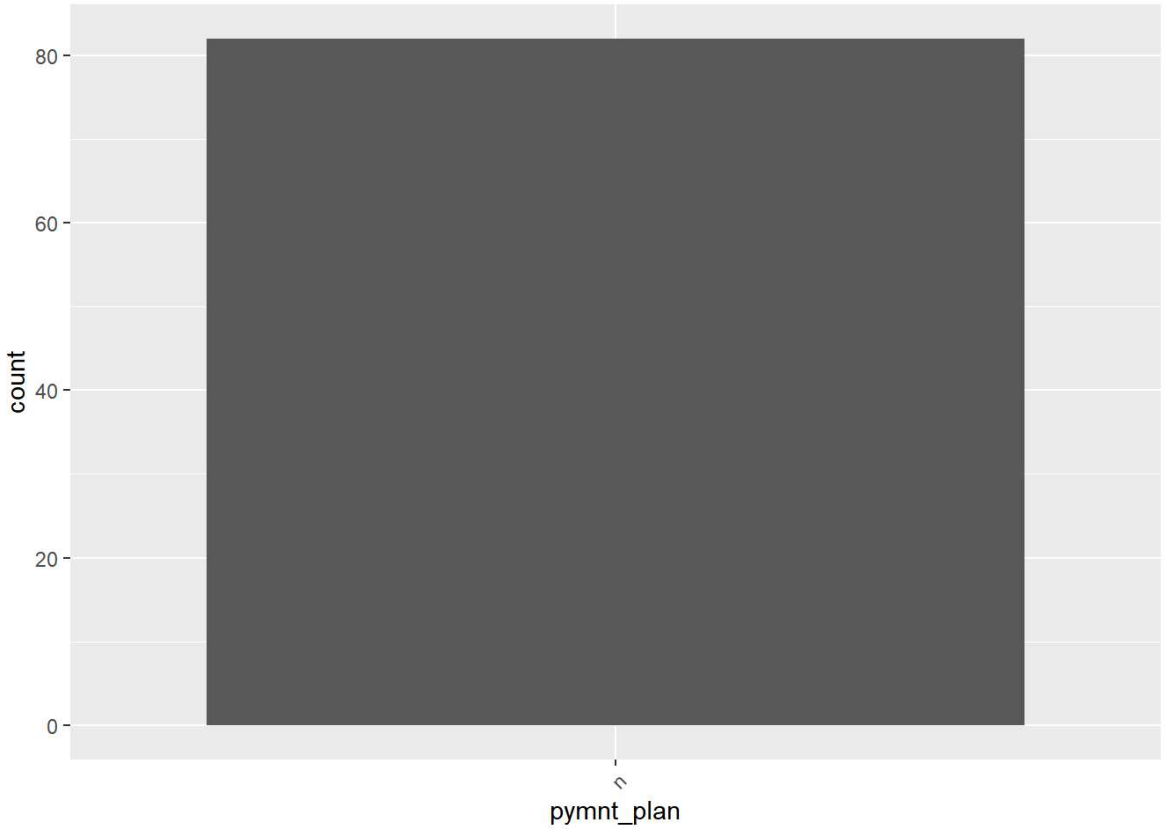

25
20
15
10
5
0
count
B
C
D
E
F
G
grade

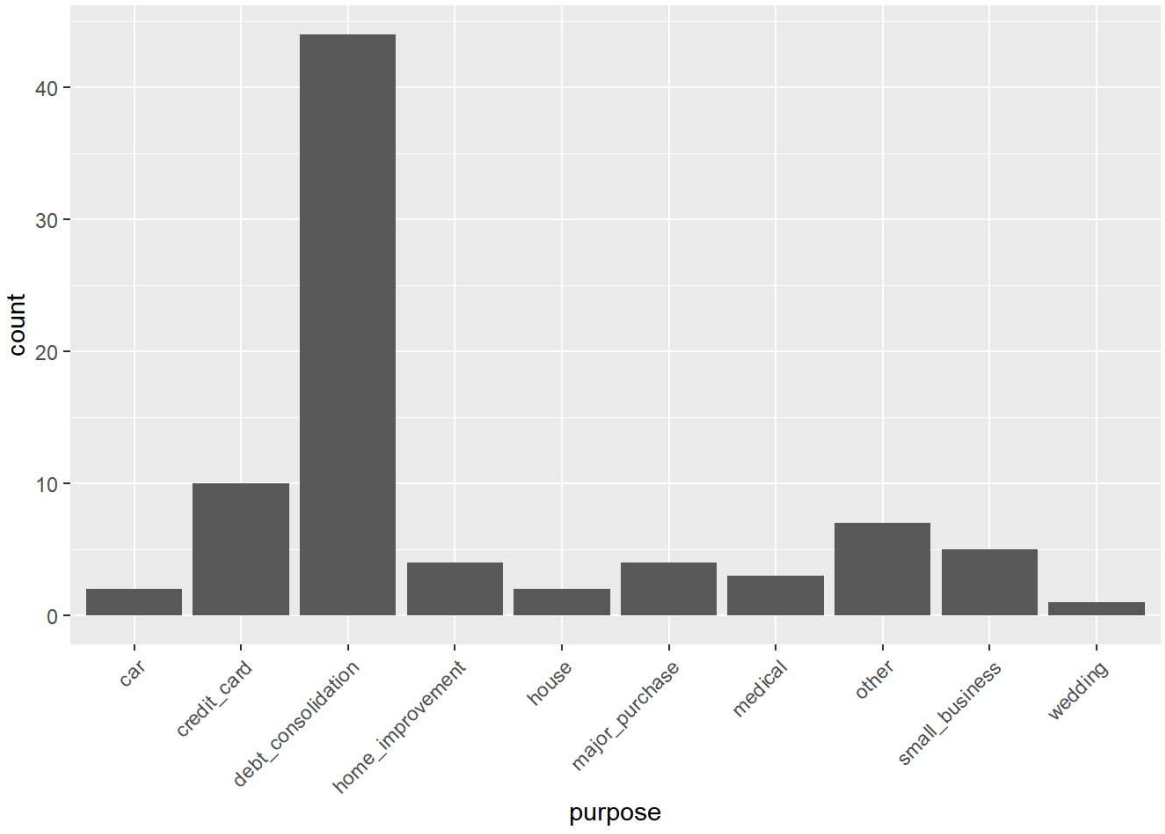

count
0
2
4
6
B4
B5
C1
C2
C3
C4
C5
D1
D2
D3
D4
D5
E1
E2
E3
E4
E5
F1
F2
F3
F4
G1
G2
G4
G5
sub_grade

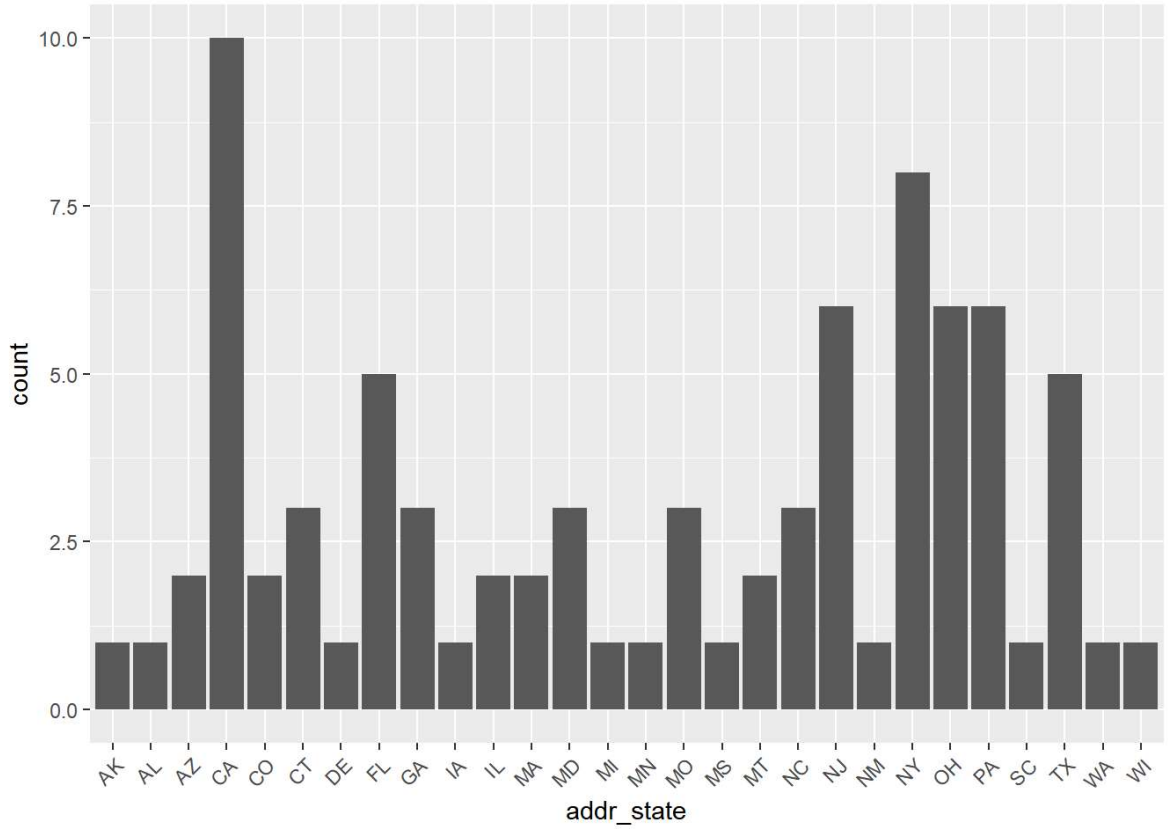

count
0
5
10
15
20
< 1 year
1 year
10+ years
2 years
3 years
4 years
5 years
6 years
7 years
8 years
9 years
emp_length

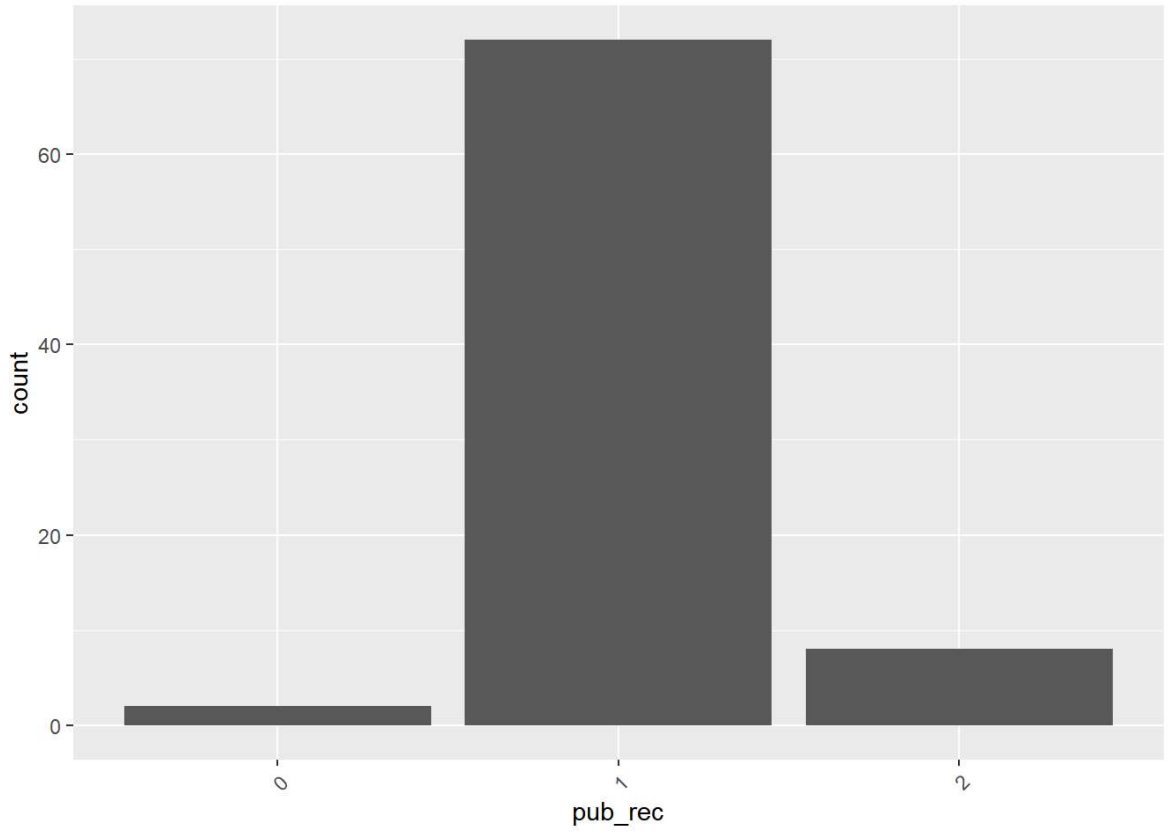

count
30
20
10
0
MORTGAGE
OTHER
OWN
RENT
home_ownership

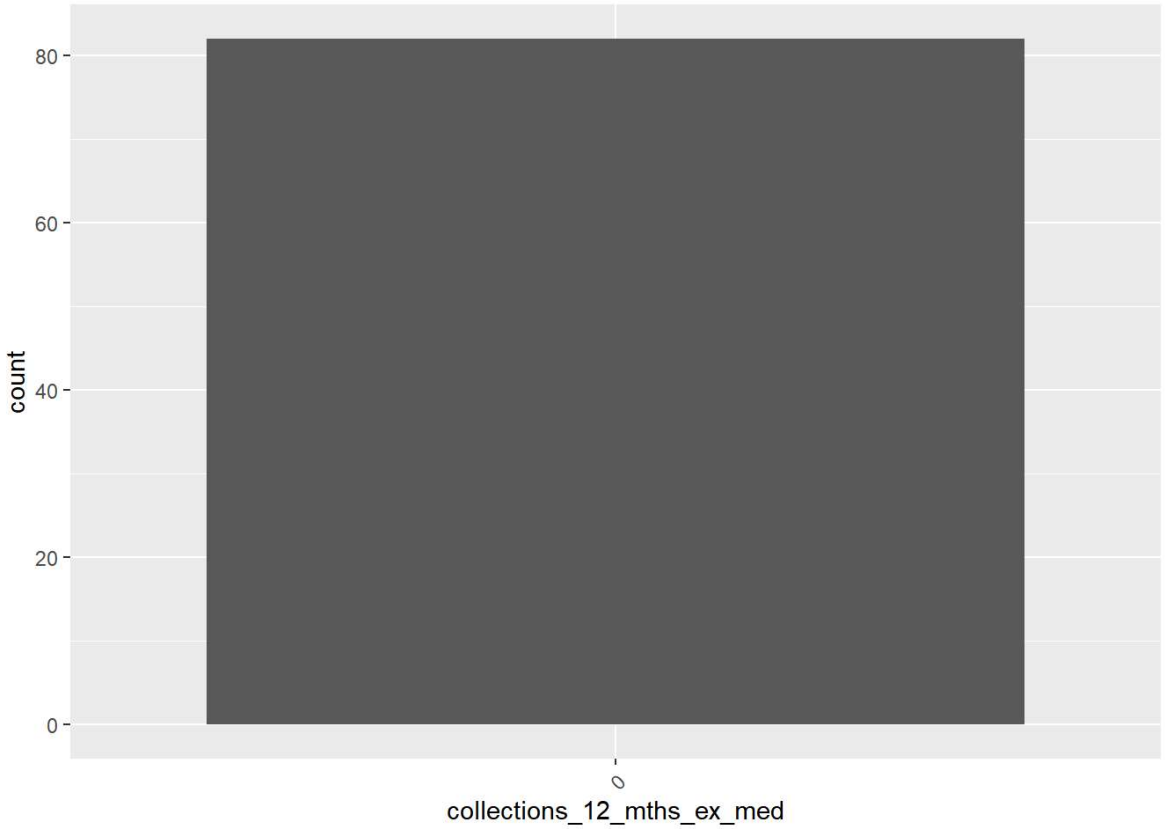

count
40
20
0
Not Verified
Source Verified
Verified
verification_status

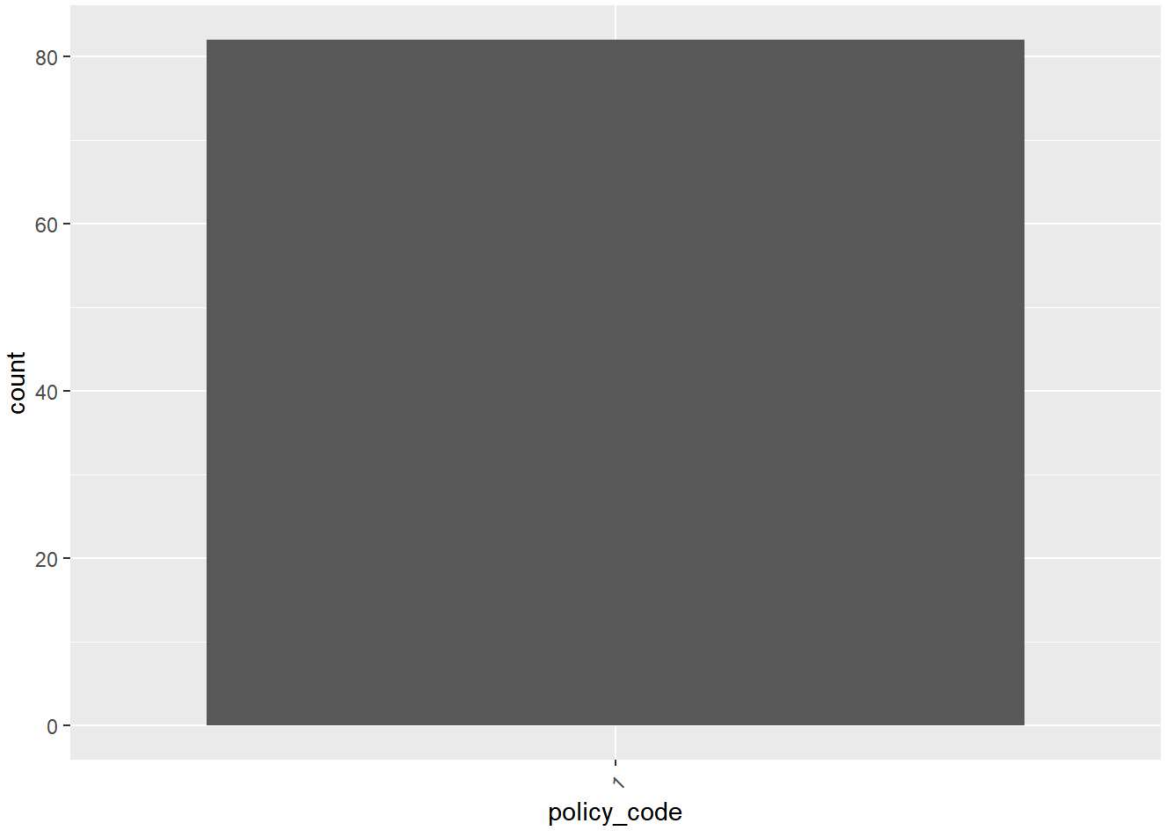

80
60
40
20
0
count
n
pymnt_plan

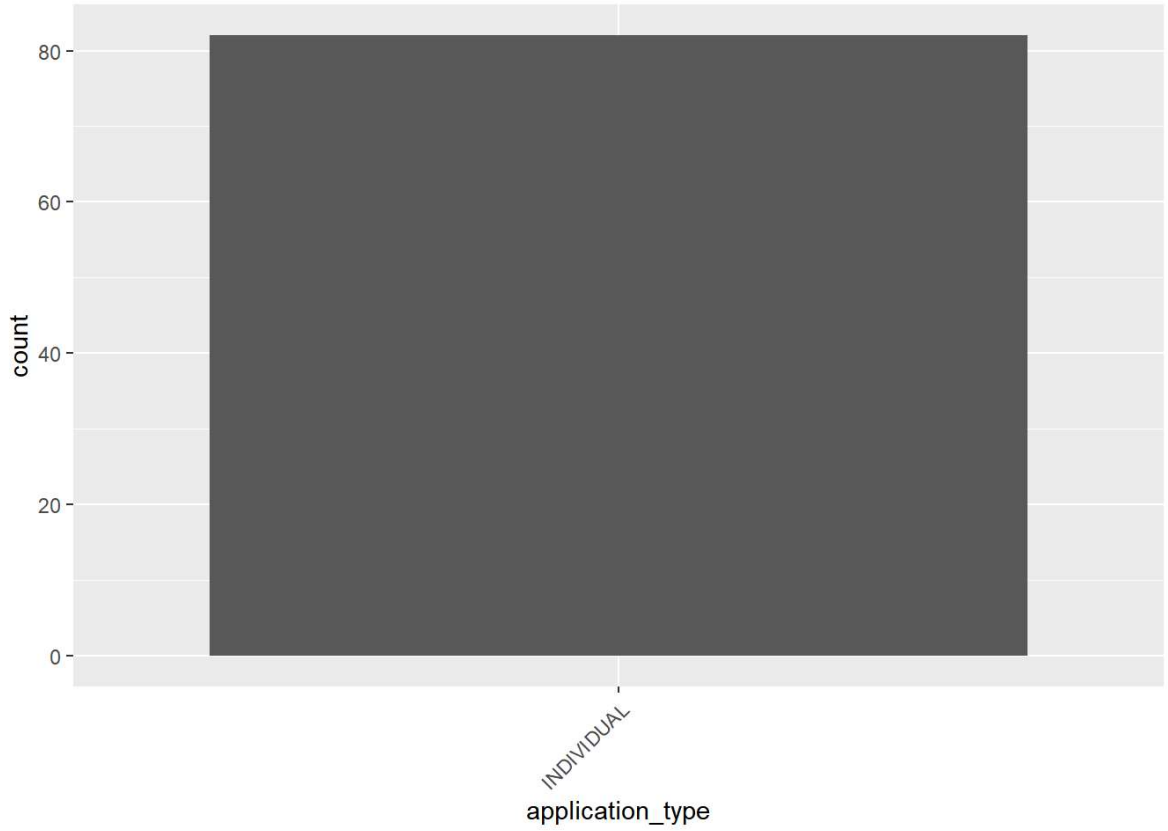

40
30
20
10
0
count
car
credit_card
debt_consolidation
home_improvement
house
major_purchase
medical
other
small_business
wedding
purpose


10.0
7.5
5.0
2.5
0.0
count
AK
AL
AZ
CA
CO
CT
DE
FL
GA
IA
IL
MA
MD
MI
MN
MO
MS
MT
NC
NJ
NM
NY
OH
PA
SC
TX
WA
WI
addr_state

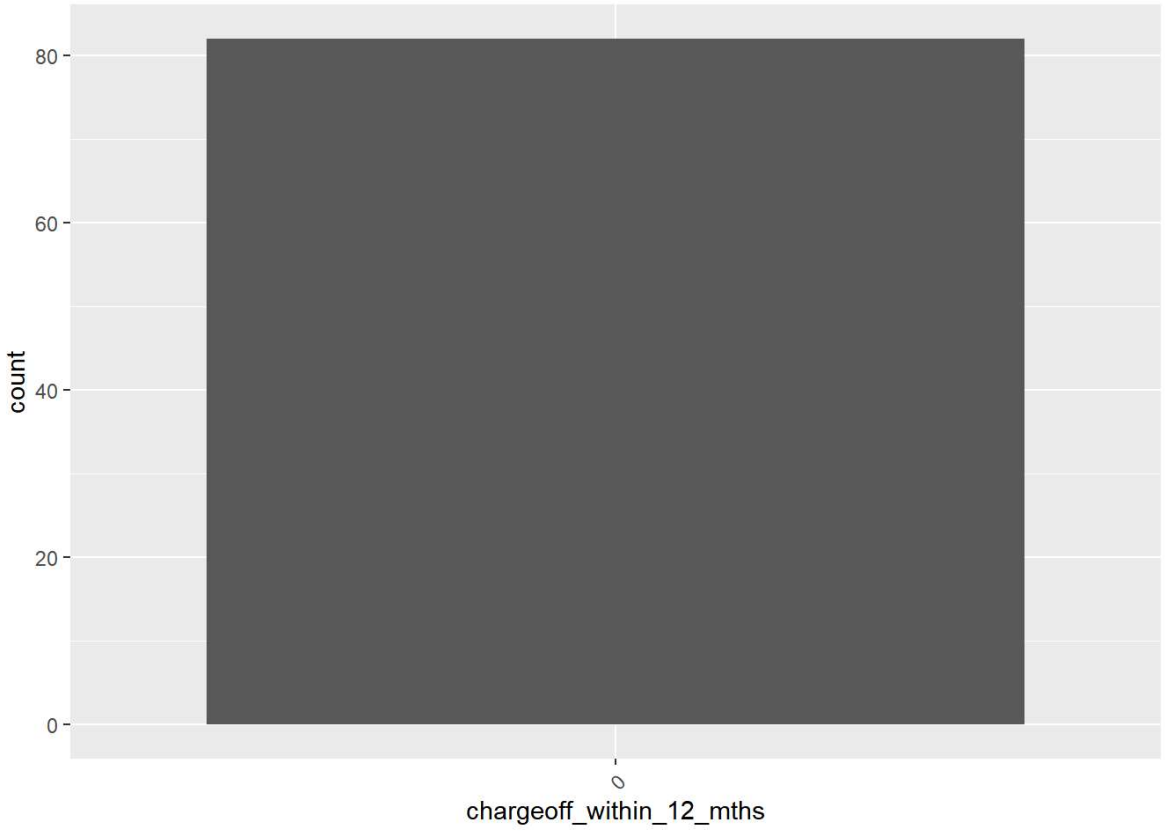

60
40
20
0
count
0
1
2
pub_rec


80
60
40
20
0
count
0
collections_12_mths_ex_med

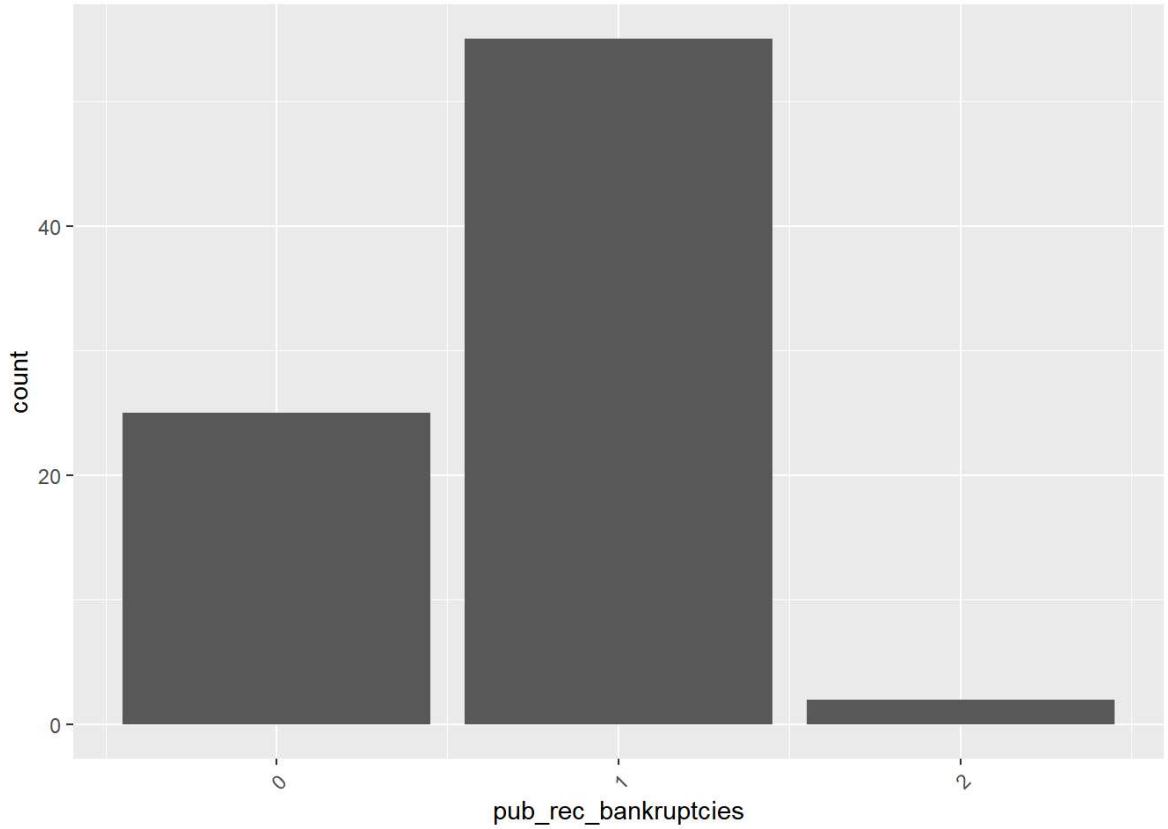

80
60
40
20
0
count
1
policy_code

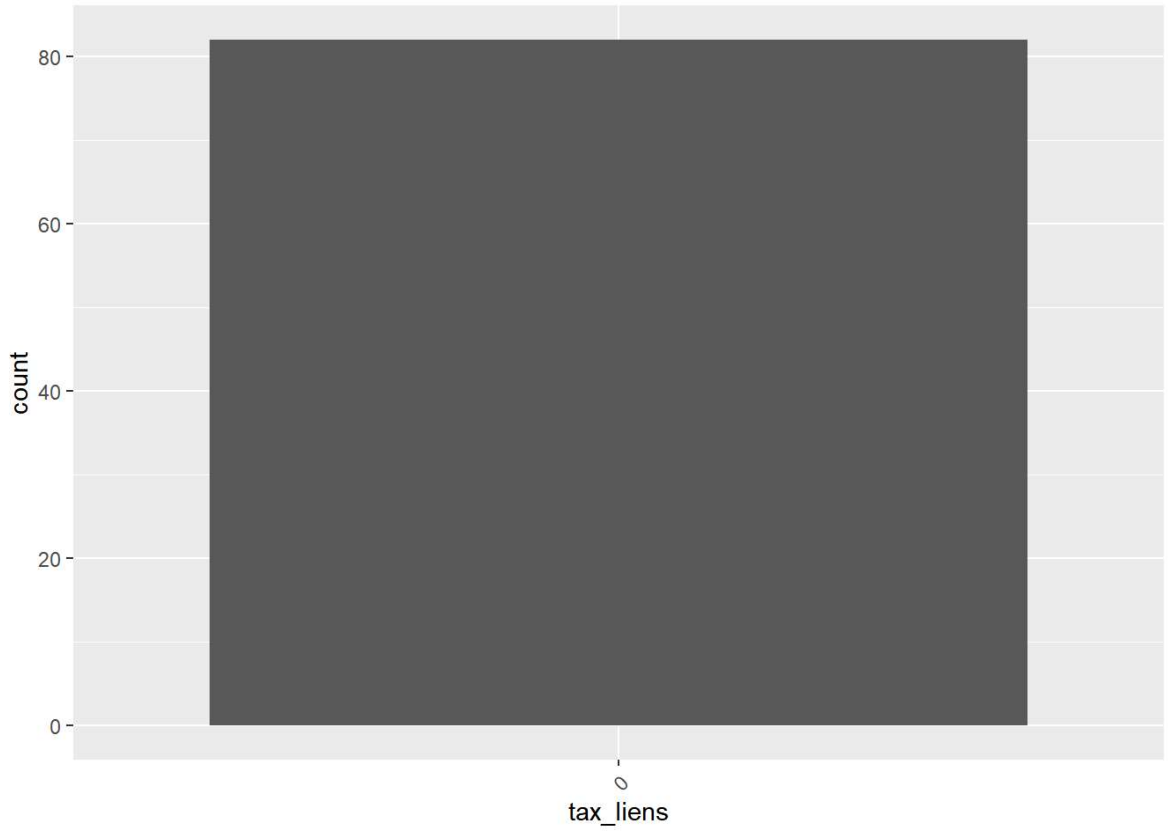

80
60
40
20
0
count
INDIVIDUAL
application_type


80
60
40
20
0
count
0
acc_now_delinq


80
60
40
20
0
count
0
chargeoff_within_12_mths


80
60
40
20
0
count
0
delinq_amnt





Correlations

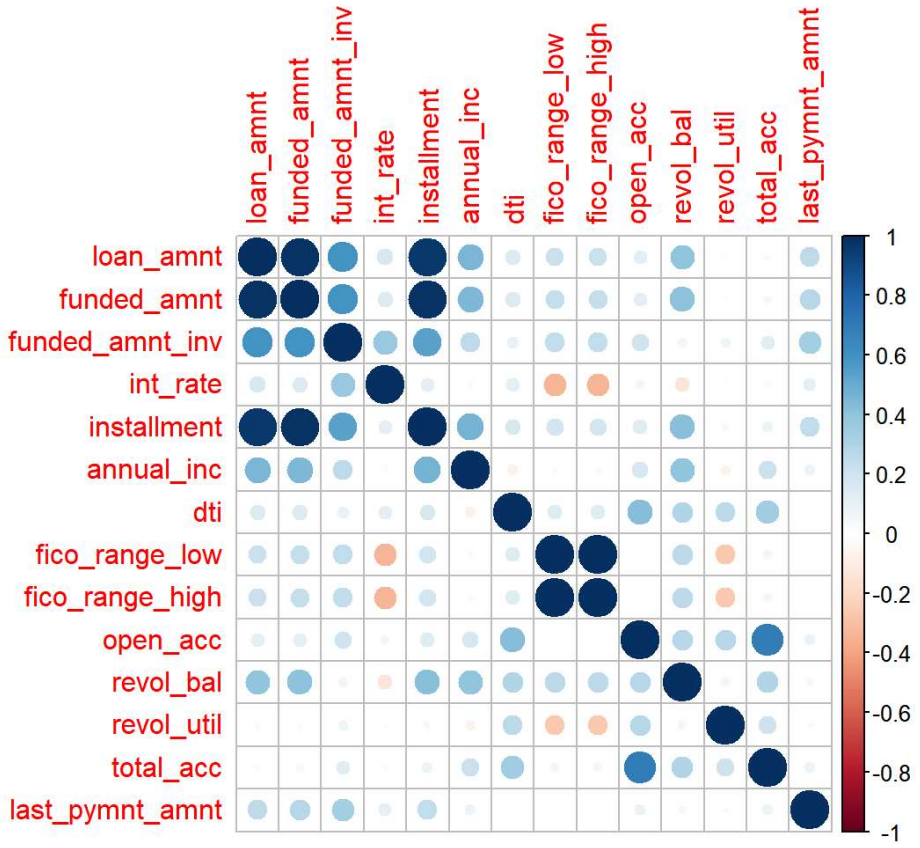
create a correlation matrix of key numeric varaibles: loan_amnt, funded_amnt, funded_amnt_inv, int_rate, installment, annual_inc, dti, fico_range_low, fico_range_high, open_acc, revol_bal, revol_util, total_acc, last_pymnt_amnt

hint: you need to deal with missing values

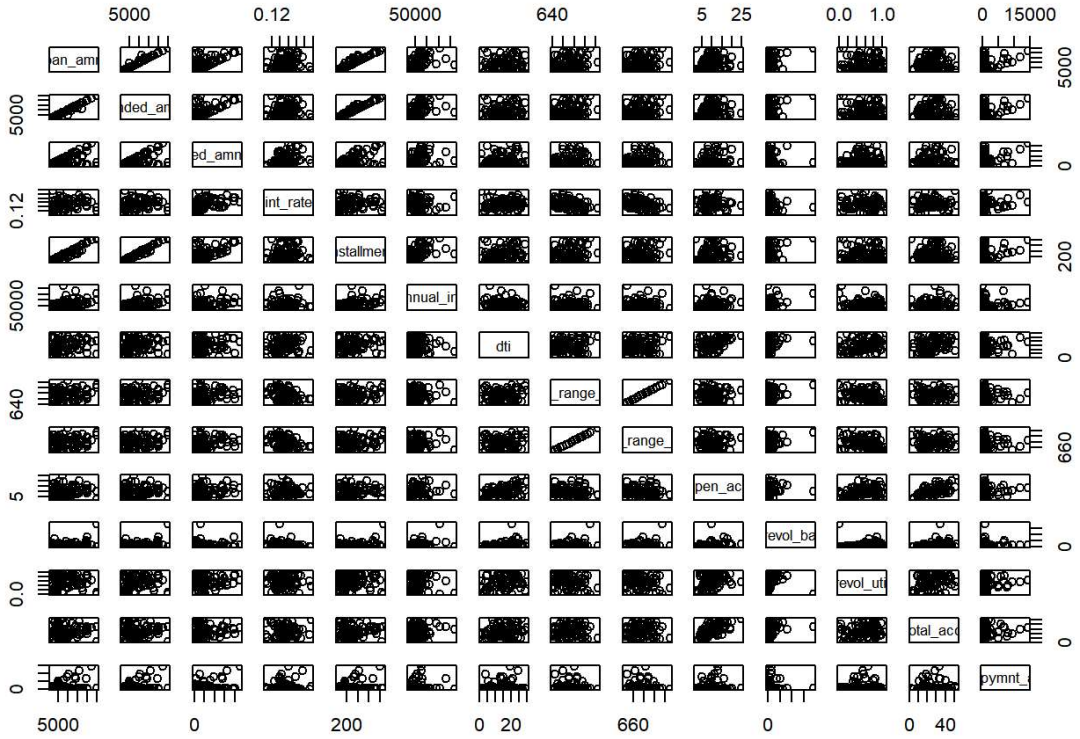
```
cor_analysis <- loan_decimal %>%
  na.omit() %>%
  dplyr::select(loan_amnt, funded_amnt, funded_amnt_inv, int_rate, installment, annual_inc, dti, fico_range_low, fico_range_h
          igh, open_acc, revol_bal, revol_util, total_acc, last_pymnt_amnt) %>%
  cor() %>%
  melt() %>% #turn it into a dataframe
  arrange(desc(value))


cor_analysis_1 <- loan_decimal %>%
  na.omit() %>%
  dplyr::select(loan_amnt, funded_amnt, funded_amnt_inv, int_rate, installment, annual_inc, dti, fico_range_low, fico_range_h
          igh, open_acc, revol_bal, revol_util, total_acc, last_pymnt_amnt)


cormat <- cor(cor_analysis_1)
round(cormat, 2)
```

```
##                loan_amnt funded_amnt funded_amnt_inv int_rate installment
## loan_amnt           1.00        0.99            0.59     0.17        0.97
## funded_amnt         0.99        1.00            0.59     0.15        0.98
## funded_amnt_inv     0.59        0.59            1.00     0.37        0.54
## int_rate            0.17        0.15            0.37     1.00        0.12
## installment         0.97        0.98            0.54     0.12        1.00
## annual_inc          0.45        0.45            0.26     0.03        0.47
## dti                 0.15        0.15            0.09     0.12        0.16
## fico_range_low      0.22        0.24            0.25    -0.33        0.19
## fico_range_high     0.22        0.24            0.25    -0.33        0.19
## open_acc            0.12        0.11            0.20     0.05        0.14
## revol_bal           0.39        0.40            0.06    -0.13        0.42
## revol_util          0.02        0.02            0.06     0.01        0.03
## total_acc           0.03        0.04            0.12     0.03        0.07
## last_pymnt_amnt     0.25        0.27            0.33     0.11        0.24
##                annual_inc   dti fico_range_low fico_range_high open_acc
## loan_amnt            0.45  0.15           0.22            0.22     0.12
## funded_amnt          0.45  0.15           0.24            0.24     0.11
## funded_amnt_inv      0.26  0.09           0.25            0.25     0.20
## int_rate             0.03  0.12          -0.33           -0.33     0.05
## installment          0.47  0.16           0.19            0.19     0.14
## annual_inc           1.00 -0.08           0.02            0.02     0.18
## dti                 -0.08  1.00           0.13            0.13     0.43
## fico_range_low       0.02  0.13           1.00            1.00     0.00
## fico_range_high      0.02  0.13           1.00            1.00     0.00
## open_acc             0.18  0.43           0.00            0.00     1.00
## revol_bal            0.39  0.29           0.27            0.27     0.27
## revol_util          -0.06  0.26          -0.26           -0.26     0.27
## total_acc            0.21  0.35           0.05            0.05     0.70
## last_pymnt_amnt      0.08  0.01           0.00            0.00     0.08
##                revol_bal revol_util total_acc last_pymnt_amnt
## loan_amnt           0.39       0.02      0.03            0.25
## funded_amnt         0.40       0.02      0.04            0.27
## funded_amnt_inv     0.06       0.06      0.12            0.33
## int_rate           -0.13       0.01      0.03            0.11
## installment         0.42       0.03      0.07            0.24
## annual_inc          0.39      -0.06      0.21            0.08
## dti                 0.29       0.26      0.35            0.01
## fico_range_low      0.27      -0.26      0.05            0.00
## fico_range_high     0.27      -0.26      0.05            0.00
## open_acc            0.27       0.27      0.70            0.08
## revol_bal           1.00       0.06      0.30            0.04
## revol_util          0.06       1.00      0.20           -0.03
## total_acc           0.30       0.20      1.00            0.08
## last_pymnt_amnt     0.04      -0.03      0.08            1.00
```

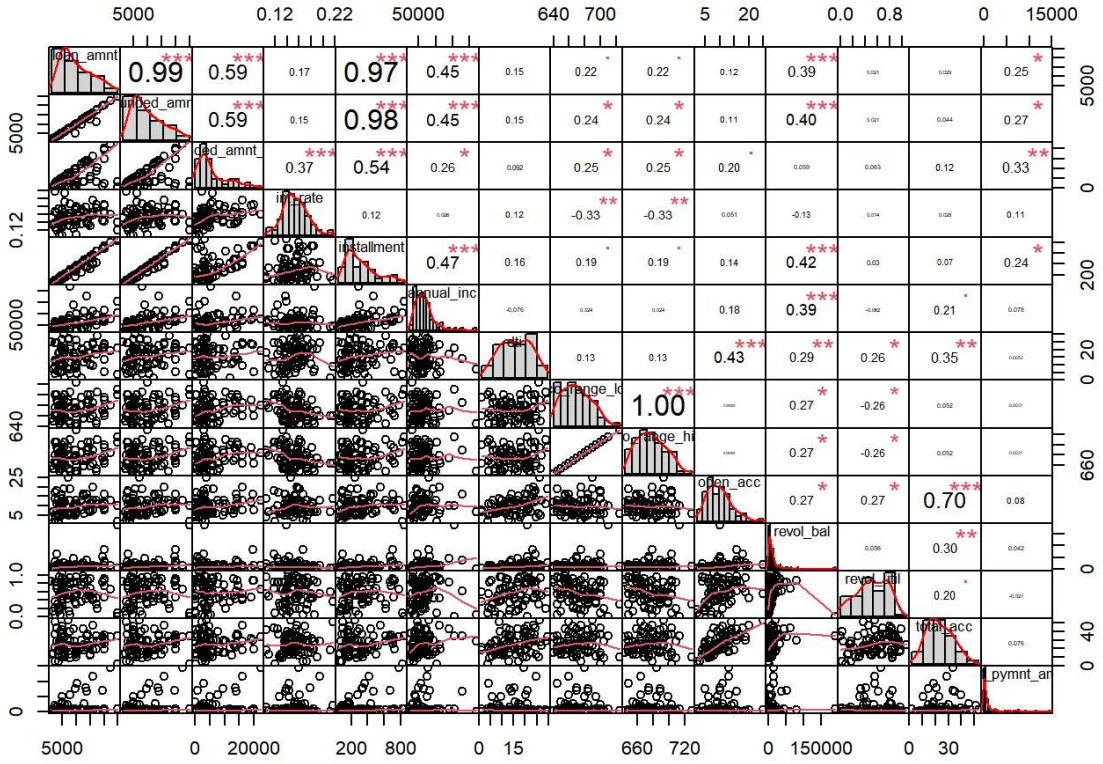
```
corrplot(cormat)
```



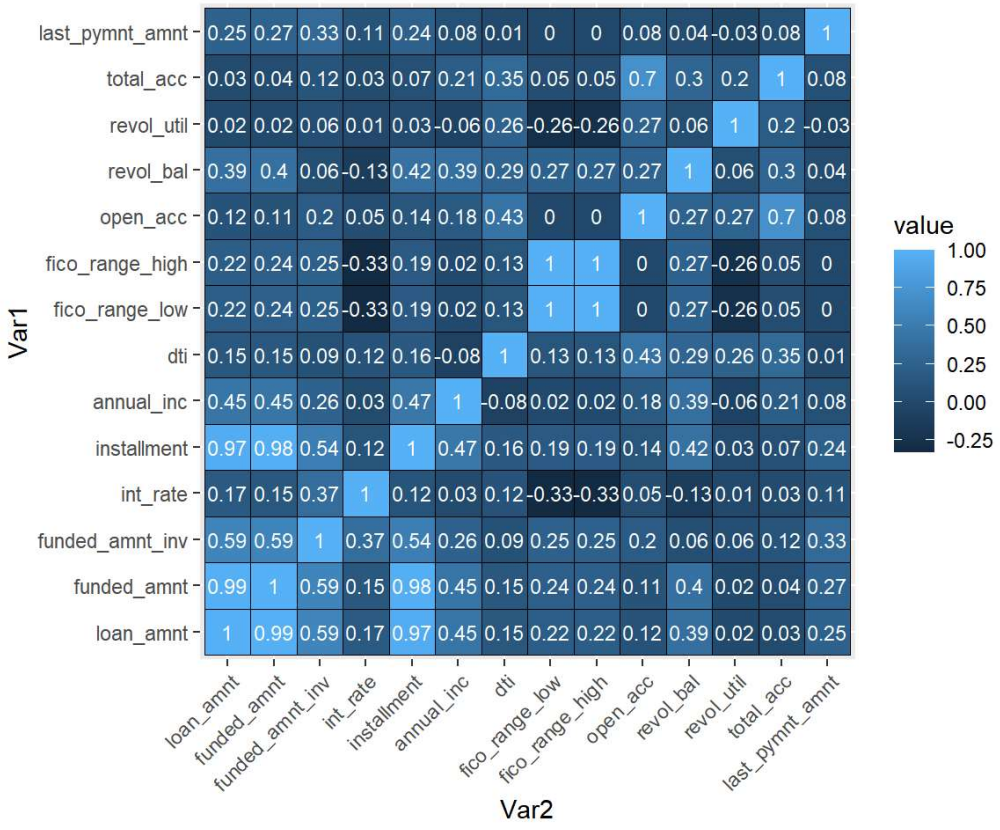

```
pairs(cor_analysis_1)
```




```
chart.Correlation(cor_analysis_1, histogram=TRUE, pch=4)
```




```
cor_analysis %>%
  ggplot(aes(Var2, Var1, fill = value)) +
  geom_tile(color = "black")+ geom_text(aes(label = round(value,2)), color = "white", size = 3) +
  coord_fixed() +
  theme(axis.text.x=element_text(angle=45, hjust=1))
```



Data Transformation

```
data_prep <- loan_decimal %>% dplyr::select(loan_status, loan_amnt, funded_amnt, funded_amnt_inv, int_rate, installment, annual_inc, dti, delinq_2yrs, fico_range_low, fico_range_high, inq_last_6mths, open_acc, revol_bal, revol_util, total_acc, total_rec_late_fee, last_pymnt_amnt, issue_d_year, earliest_cr_line_year, last_pymnt_d_year, last_credit_pull_d_year, term, grade, sub_grade, emp_length, home_ownership, verification_status, purpose, addr_state, pub_rec, tax_liens) %>%
    mutate(loan_status = as.factor(loan_status)) %>%
    mutate_if(is.character,factor)

head(data_prep)
```

loan_status <fct>	loan_amnt <dbl>	funded_amnt <dbl>	funded_amnt_inv <dbl>	int_rate <dbl>	installment <dbl>	annual_inc <dbl>	dti <dbl>	delinq_2yrs <dbl>
current	5000	5000	4975	0.1065	162.87	24000	27.65	0
default	2500	2500	2500	0.1527	59.83	30000	1.00	0
current	10000	10000	10000	0.1349	339.31	49200	20.00	0
current	7000	7000	7000	0.1596	170.08	47004	23.51	0
current	3000	3000	3000	0.1864	109.43	48000	5.35	0
default	5600	5600	5600	0.2128	152.39	40000	5.55	0

6 rows | 1-9 of 32 columns

```
data_prep %>% skim()
```

Data summary

Name	Piped data
Number of rows	29777
Number of columns	32
_____	
Column type frequency:	
factor	11
numeric	21
_____	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
loan_status	0	1	FALSE	2	cur: 25300, def: 4477
term	3	1	FALSE	2	36 : 22160, 60 : 7614
grade	3	1	FALSE	7	B: 8620, A: 7142, C: 6068, D: 4268
sub_grade	3	1	FALSE	35	B3: 2088, A4: 2044, A5: 1957, B5: 1932
emp_length	3	1	FALSE	12	10+: 6577, < 1: 3491, 2 y: 3313, 3 y: 3008
home_ownership	3	1	FALSE	5	REN: 14064, MOR: 13340, OWN: 2275, OTH: 91
verification_status	3	1	FALSE	3	Not: 13128, Ver: 9460, Sou: 7186

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
purpose	3	1	FALSE	14	deb: 13816, cre: 3850, oth: 3108, hom: 2226
addr_state	3	1	FALSE	50	CA: 5188, NY: 2836, FL: 2200, TX: 2067
pub_rec	23	1	FALSE	6	0: 28086, 1: 1610, 2: 45, 3: 11
tax_liens	79	1	FALSE	2	0: 29697, 1: 1

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
loan_amnt	3	1	11110	9.43	7404.65	500.00	5225.00	9800.00	15000.00	35000.00
funded_amnt	3	1	10843	9.64	7147.05	500.00	5100.00	9600.00	15000.00	35000.00
funded_amnt_inv	3	1	10149	9.66	7130.86	0.00	4950.00	8500.00	14000.00	35000.00
int_rate	3	1	0.12	0.04	0.05	0.10	0.12	0.15	0.24	
installment	3	1	323.81	209.77	15.67	165.84	278.94	429.86	1305.19	
annual_inc	4	1	69201.23	66566.42	2000.00	40000.00	59000.00	82500.00	6000000.00	
dti	3	1	13.38	6.74	0.00	8.19	13.49	18.70	29.99	
delinq_2yrs	23	1	0.16	0.52	0.00	0.00	0.00	0.00	13.00	
fico_range_low	3	1	713.05	36.31	610.00	685.00	710.00	740.00	825.00	
fico_range_high	3	1	717.05	36.31	614.00	689.00	714.00	744.00	829.00	
inq_last_6mths	23	1	1.08	1.54	0.00	0.00	1.00	2.00	33.00	
open_acc	23	1	9.34	4.51	1.00	6.00	9.00	12.00	47.00	
revol_bal	3	1	14310.00	22696.55	0.00	3644.00	8783.50	17187.00	1207359.00	
revol_util	67	1	0.49	0.28	0.00	0.26	0.50	0.73	1.19	
total_acc	23	1	22.08	11.59	1.00	13.00	20.00	29.00	81.00	
total_rec_late_fee	3	1	1.50	7.72	0.00	0.00	0.00	0.00	180.20	
last_pymnt_amnt	3	1	2615.41	4373.70	0.00	211.02	531.98	3171.29	36115.20	
issue_d_year	3	1	11.78	0.97	11.00	11.00	11.00	12.00	15.00	
earliest_cr_line_year	23	1	25.43	6.85	14.00	21.00	24.00	29.00	76.00	
last_pymnt_d_year	67	1	9.30	1.61	6.00	8.00	9.00	10.00	15.00	
last_credit_pull_d_year	5	1	7.43	1.81	6.00	6.00	6.00	9.00	15.00	

Remove Anomaly Records (Isolation Forest)

Recipe & Bake for Isolation Forest

```
# deal w. categoricals
loan_recipe <- recipe(~.,data_prep) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_novel(all_nominal_predictors()) %>%
  step_impute_median(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  prep()

bake_loan <- bake(loan_recipe, data_prep)

# just numeric data
loan_numeric <- data_prep %>% select_if(is.numeric)

loan_recipe <- recipe(~.,loan_numeric) %>%
  step_impute_median(all_numeric()) %>%
  prep()
```

Train My IsolationForest

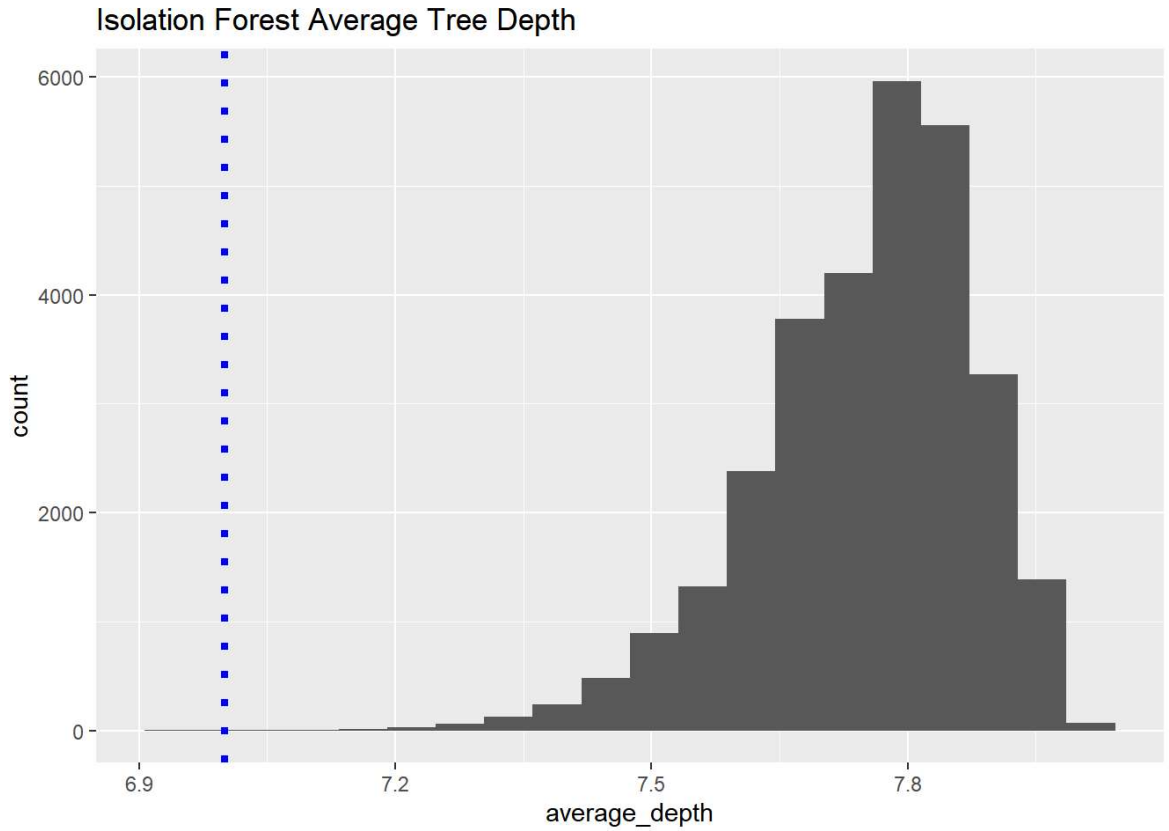
```
iso_forest <- isolationForest$new(
  sample_size = 256,
  num_trees = 100,
  max_depth = ceiling(log2(256)))

iso_forest$fit(bake_loan)
```

```
## INFO  [18:53:45.796] dataset has duplicated rows
## INFO  [18:53:45.844] Building Isolation Forest ...
## INFO  [18:53:46.484] done
## INFO  [18:53:46.485] Computing depth of terminal nodes ...
## INFO  [18:53:47.134] done
## INFO  [18:53:48.232] Completed growing isolation forest
```

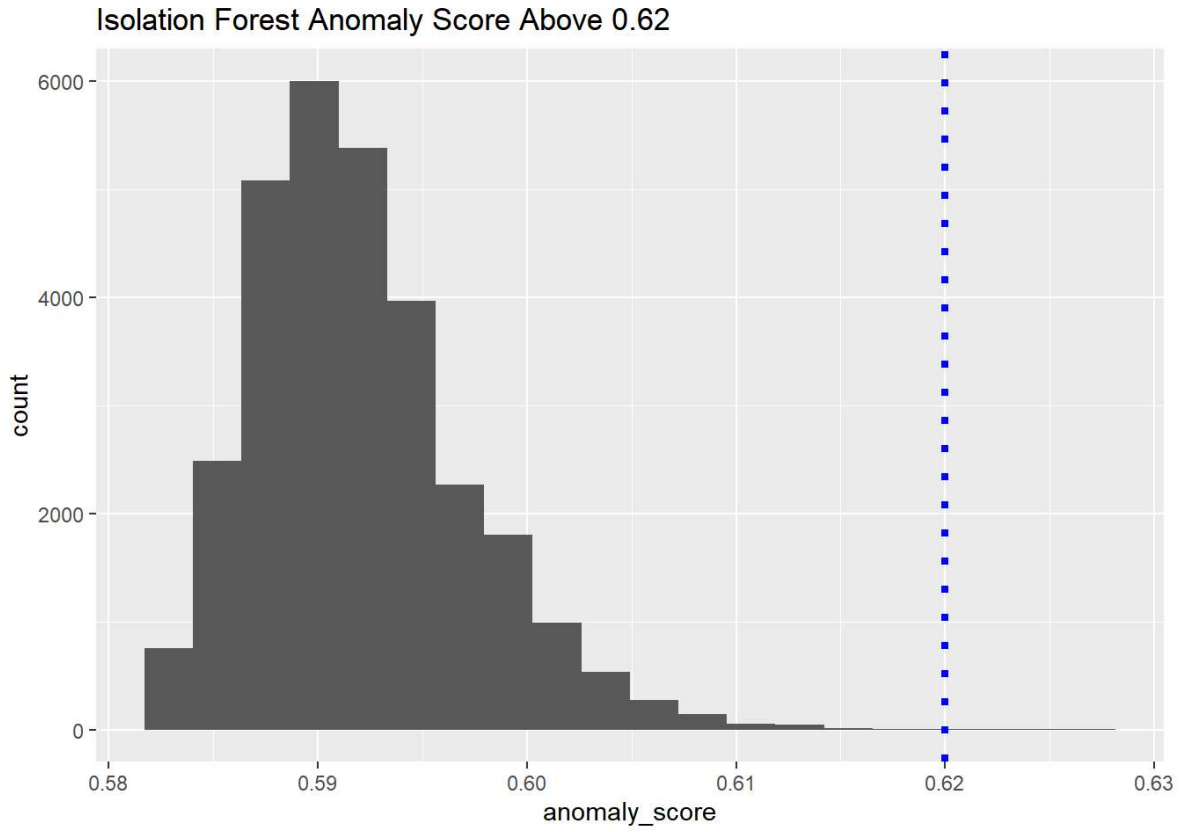
Predict Training

evaluate histogram pick a value of average_depth to identify anomalies. a shorter average depth means the point is more isolated and more likely an anomaly

```
pred_train <- iso_forest$predict(bake_loan)

pred_train %>%
  ggplot(aes(average_depth)) +
  geom_histogram(bins=20) +
  geom_vline(xintercept = 7, linetype="dotted",
                color = "blue", size=1.5) +
  labs(title="Isolation Forest Average Tree Depth")
```




```
pred_train %>%
  ggplot(aes(anomaly_score)) +
  geom_histogram(bins=20) +
  geom_vline(xintercept = 0.62, linetype="dotted",
             color = "blue", size=1.5) +
  labs(title="Isolation Forest Anomaly Score Above 0.62")
```



Global Level Interpretation

The steps of interpreting anomalies on a global level are:

1. Create a data frame with a column that indicates whether the record was considered an anomaly.
2. Train a decision tree to predict the anomaly flag.
3. Visualize the decision tree to determine which segments of the data are considered anomalous.

```
train_pred <- bind_cols(iso_forest$predict(bake_loan),bake_loan) %>%
  mutate(anomaly = as.factor(if_else(average_depth <= 7.1, "Anomaly","Normal")))

train_pred %>%
  arrange(average_depth) %>%
  count(anomaly)
```

anomaly	n
<fct>	<int>
Anomaly	5
Normal	29772

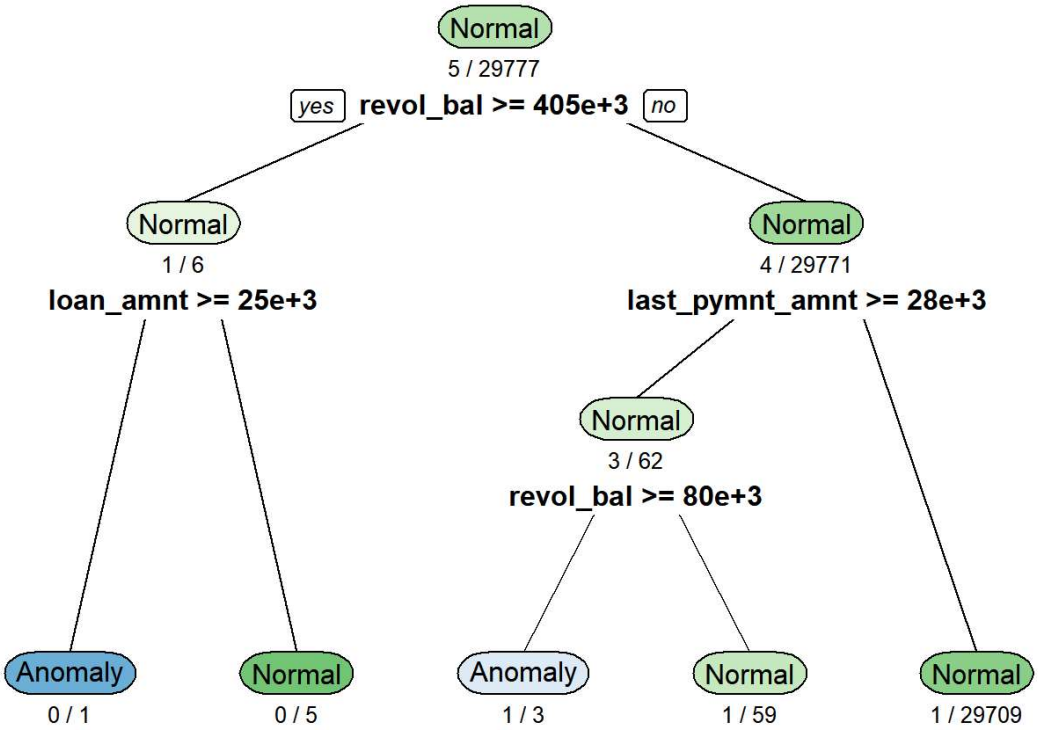
2 rows

Fit a Tree

```
fmla <- as.formula(paste("anomaly ~ ", paste(bake_loan %>% colnames(), collapse= "+")))

outlier_tree <- decision_tree(min_n=2, tree_depth=3, cost_complexity = .01) %>%
  set_mode("classification") %>%
  set_engine("rpart") %>%
  fit(fmla, data=train_pred)

outlier_tree$fit
```

```
## n= 29777
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 29777 5 Normal (0.00016791483 0.99983208517)
##    2) revol_bal>=404867.5 6 1 Normal (0.16666666667 0.83333333333)
##      4) loan_amnt>=24500 1 0 Anomaly (1.00000000000 0.00000000000) *
##      5) loan_amnt< 24500 5 0 Normal (0.00000000000 1.00000000000) *
##    3) revol_bal< 404867.5 29771 4 Normal (0.00013435894 0.99986564106)
##      6) last_pymnt_amnt>=27804.08 62 3 Normal (0.04838709677 0.95161290323)
##       12) revol_bal>=79776.5 3 1 Anomaly (0.66666666667 0.33333333333) *
##       13) revol_bal< 79776.5 59 1 Normal (0.01694915254 0.98305084746) *
##      7) last_pymnt_amnt< 27804.08 29709 1 Normal (0.00003365983 0.99996634017) *
```

```
library(rpart.plot) # -- plotting decision trees

rpart.plot(outlier_tree$fit,clip.right.labs = FALSE, branch = .3, under = TRUE, roundint=FALSE, extra=3)
```



Global Anomaly Rules

```
anomaly_rules <- rpart.rules(outlier_tree$fit,roundint=FALSE, extra = 4, cover = TRUE, clip.facs = TRUE) %>% clean_names() %
          >%
  #filter(anomaly=="Anomaly") %>%
  mutate(rule = "IF")



rule_cols <- anomaly_rules %>% dplyr::select(starts_with("x_")) %>% colnames()

for (col in rule_cols){
anomaly_rules <- anomaly_rules %>%
    mutate(rule = paste(rule, !!as.name(col)))
}

anomaly_rules %>%
  as.data.frame() %>%
  filter(anomaly == "Anomaly") %>%
  mutate(rule = paste(rule, " THEN ", anomaly )) %>%
  mutate(rule = paste(rule," coverage ", cover)) %>%
  dplyr::select( rule)
```

	rule
	<chr>
4	IF revol_bal >= 404868 & loan_amnt >= 24500 THEN Anomaly coverage 0%
12	IF revol_bal is 79777 to 404868 & last_pymnt_amnt >= 27804 THEN Anomaly coverage 0%

2 rows

```
anomaly_rules %>%
  as.data.frame() %>%
  filter(anomaly == "Normal") %>%
  mutate(rule = paste(rule, " THEN ", anomaly )) %>%
  mutate(rule = paste(rule," coverage ", cover)) %>%
  dplyr::select( rule)
```

	rule
	<chr>
13	IF revol_bal < 79777 & last_pymnt_amnt >= 27804 THEN Normal coverage 0%
7	IF revol_bal < 404868 & last_pymnt_amnt < 27804 THEN Normal coverage 100%
5	IF revol_bal >= 404868 & loan_amnt < 24500 THEN Normal coverage 0%

3 rows

Five Most Anomalous Records

```
pred_train <- bind_cols(iso_forest$predict(bake_loan),
                        bake_loan)

pred_train %>%
  arrange(desc(anomaly_score)) %>%
  slice_max(order_by=anomaly_score,n=5)
```

id	average_depth	anomaly_score	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	installment	annual_inc	
<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<d
5259	6.92	0.6261297	35000	35000	31975.0000	0.2030	933.14	543000	8.
29658	6.95	0.6248601	25000	25000	2450.0028	0.1533	870.68	165000	13
2737	6.99	0.6231713	35000	35000	34750.0000	0.2235	973.64	240000	3.
2658	7.07	0.6198074	35000	35000	34977.3467	0.2167	960.11	250000	14
29546	7.08	0.6193882	25000	25000	474.9995	0.1375	851.41	320000	22

5 rows | 1-10 of 173 columns

```
# Anomalous Records id: 2658, 2737, 5259, 29546, 29658
```

Remove Anomalous Records

```
data <- data_prep[-c(2658, 2737, 5259, 29546, 29658),]
```

Partition My Data into 70/30 Train/Test Split

```
set.seed(1234)

# -- 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%"
```

Logistic Regression

Standard Logistic Model (Full Model)

1. make a recipe

- specify a formula
- normalize (center and scale) the numeric variables - required for lasso/ridge
- dummy encode nominal predictors

```
loan_recipe <- recipe(loan_status ~ .,
                      data = train) %>%
  step_novel(all_nominal_predictors()) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_impute_median(all_numeric_predictors()) %>%
  step_impute_mode(all_nominal_predictors()) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors(), one_hot = TRUE)

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

logistic_wf <- workflow() %>%
  add_recipe(loan_recipe) %>%
  add_model(logistic_spec) %>%
  fit(train)

logistic_wf %>%
  pull_workflow_fit() %>%
  tidy() %>%
  mutate(across(is.numeric,round,3))
```

term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <dbl>
(Intercept)	-8144753408981.146	19496922883159.656	-0.418	0.676
loan_amnt	0.311	0.209	1.489	0.137
funded_amnt	0.954	0.364	2.617	0.009
funded_amnt_inv	-0.606	0.128	-4.721	0.000
int_rate	0.640	0.164	3.903	0.000
installment	-0.154	0.255	-0.605	0.545
annual_inc	-0.228	0.054	-4.175	0.000
dti	0.082	0.040	2.065	0.039
delinq_2yrs	0.022	0.036	0.625	0.532
fico_range_low	-0.165	0.079	-2.075	0.038

1-10 of 178 rows Previous **1** 2 3 4 5 6 ... 18 Next

```
predict(logistic_wf, train, type="prob") %>%
  bind_cols(predict(logistic_wf, train, type="class")) %>%
  bind_cols(train) -> logistic_train

predict(logistic_wf, test, type="prob") %>%
  bind_cols(predict(logistic_wf, test, type="class")) %>%
  bind_cols(test) -> logistic_test

logistic_train %>%
  metrics(loan_status, estimate = .pred_class, .pred_default) %>%
  mutate(part="training") %>%
bind_rows(logistic_test %>%
  metrics(loan_status, estimate = .pred_class, .pred_default) %>%
  mutate(part="testing"))
```

.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
accuracy	binary	0.94380998	training
kap	binary	0.76954903	training
mn_log_loss	binary	7.19492214	training
roc_auc	binary	0.03046805	training
accuracy	binary	0.94581281	testing
kap	binary	0.77407365	testing
mn_log_loss	binary	7.30194432	testing
roc_auc	binary	0.03179570	testing

8 rows

Logistic Model Evaluation (Full Model)

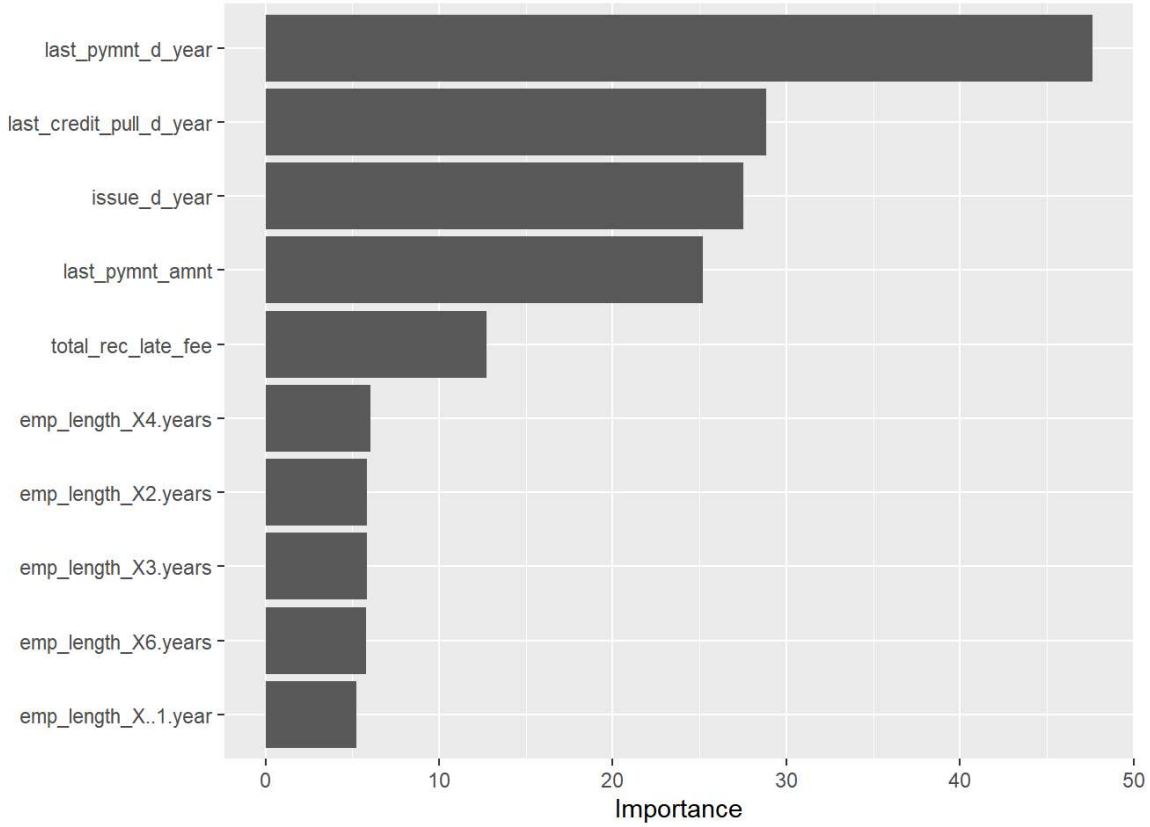
```
options(yardstick.event_first = FALSE)

# Variable Importance
logistic_wf %>%
  extract_fit_parsnip() %>%
  vi()
```

Variable	Importance	Sign
<chr>	<dbl>	<chr>
last_pymnt_d_year	47.62265460903	POS
last_credit_pull_d_year	28.83401052931	NEG
issue_d_year	27.51213574931	NEG
last_pymnt_amnt	25.19133620546	NEG

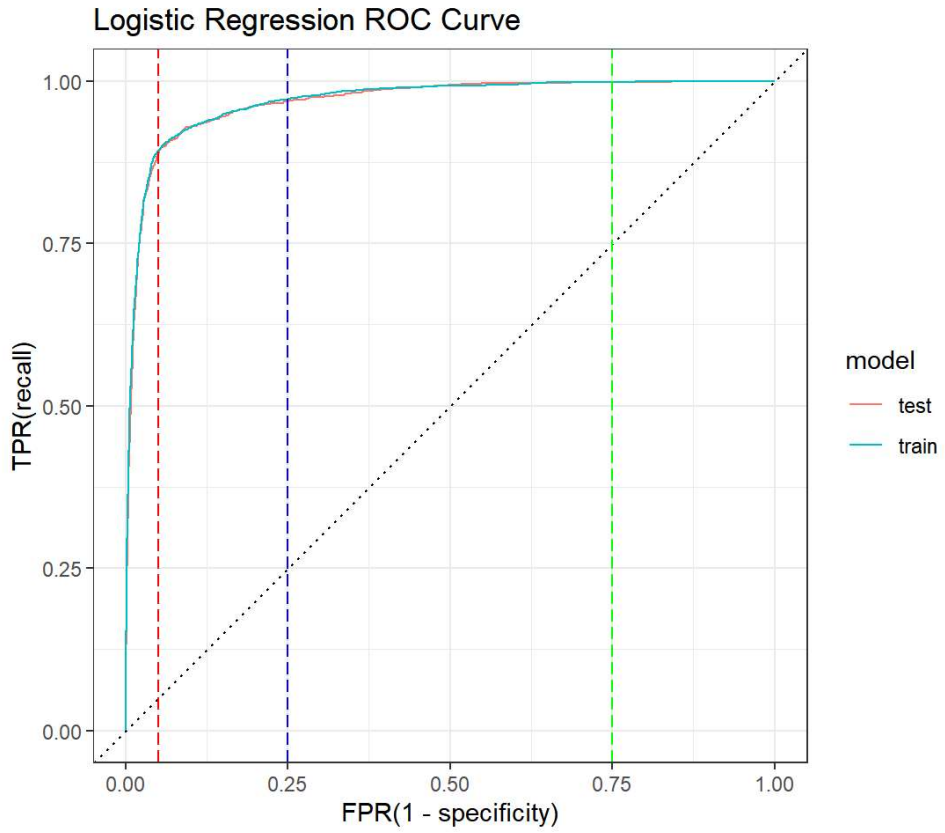
Variable	Importance	Sign
<chr>	<dbl>	<chr>
total_rec_late_fee	12.73637319101	POS
emp_length_X4.years	6.03885229142	NEG
emp_length_X2.years	5.85951572660	NEG
emp_length_X3.years	5.84634714792	NEG
emp_length_X6.years	5.78849457633	NEG
emp_length_X..1.year	5.23748694296	NEG

1-10 of 143 rows Previous **1** 2 3 4 5 6 ... 15 Next

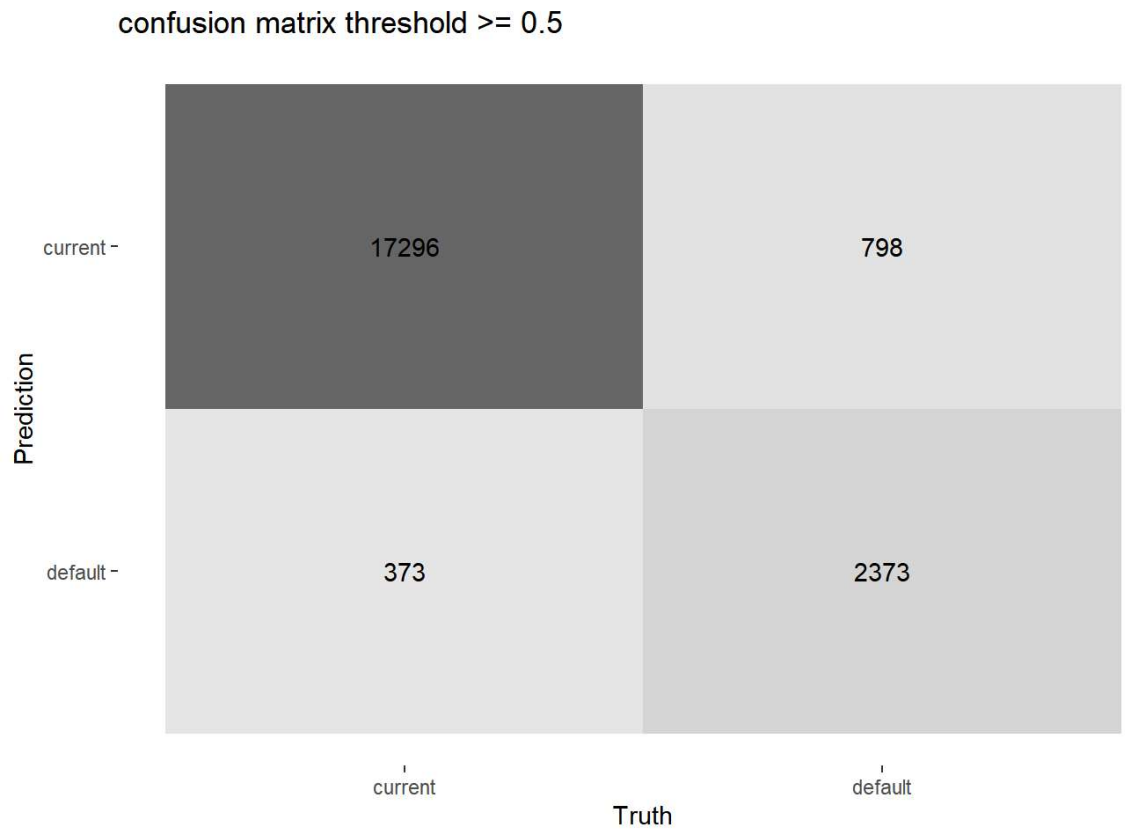
```
logistic_wf %>%
  extract_fit_parsnip() %>%
  vip()
```



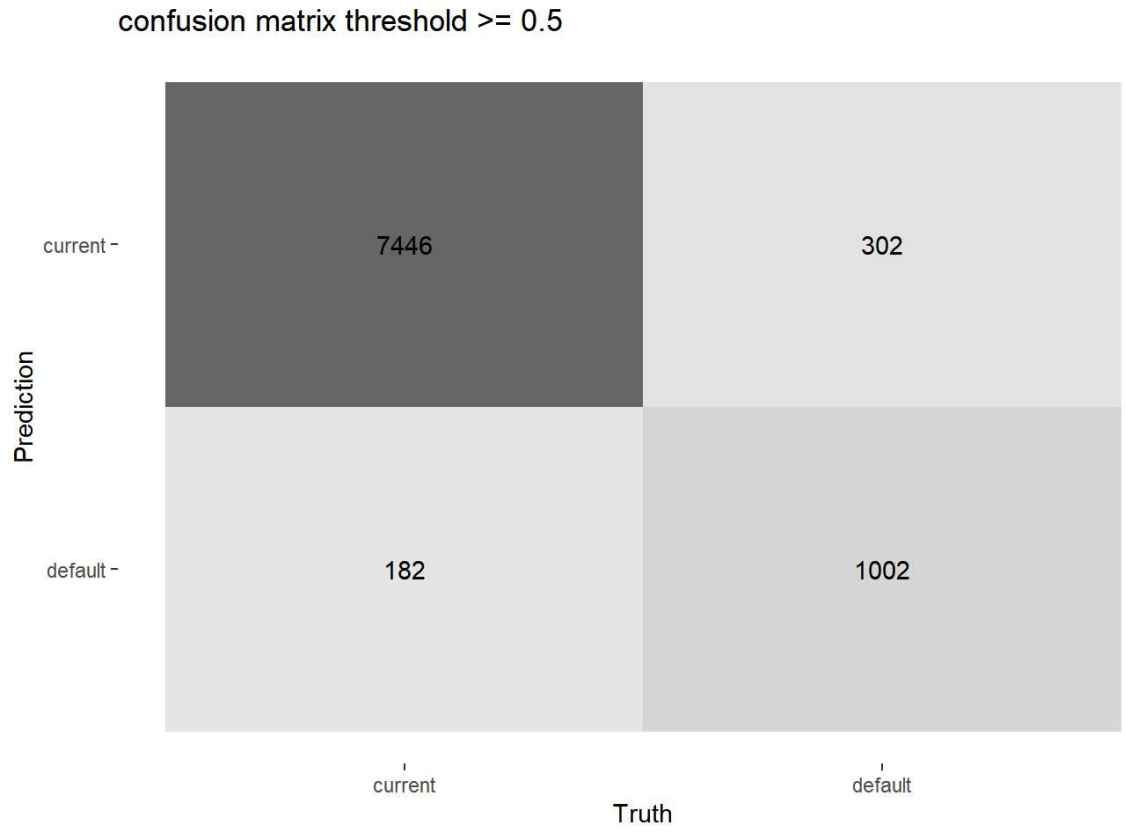
```
logistic_train <- logistic_train %>% mutate(part = "training")

logistic_test <- logistic_test %>% mutate(part = "testing")

logistic_train %>% mutate(model = "train") %>%
  bind_rows(logistic_test %>% mutate(model="test")) %>%
  group_by(model) %>%
  roc_curve(loan_status, .pred_default) %>%
  autoplot() +
  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 = "Logistic Regression ROC Curve" , x = "FPR(1 - specificity)", y = "TPR(recall)")
```




```
logistic_train %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current'))) %>%
   conf_mat(loan_status, estimate = predict_class) %>%
   autoplot(type = "heatmap") +
   labs(title="confusion matrix threshold >= 0.5")
```




```
logistic_test %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current'))) %>%
   conf_mat(loan_status, estimate = predict_class) %>%
   autoplot(type = "heatmap") +
   labs(title="confusion matrix threshold >= 0.5")
```



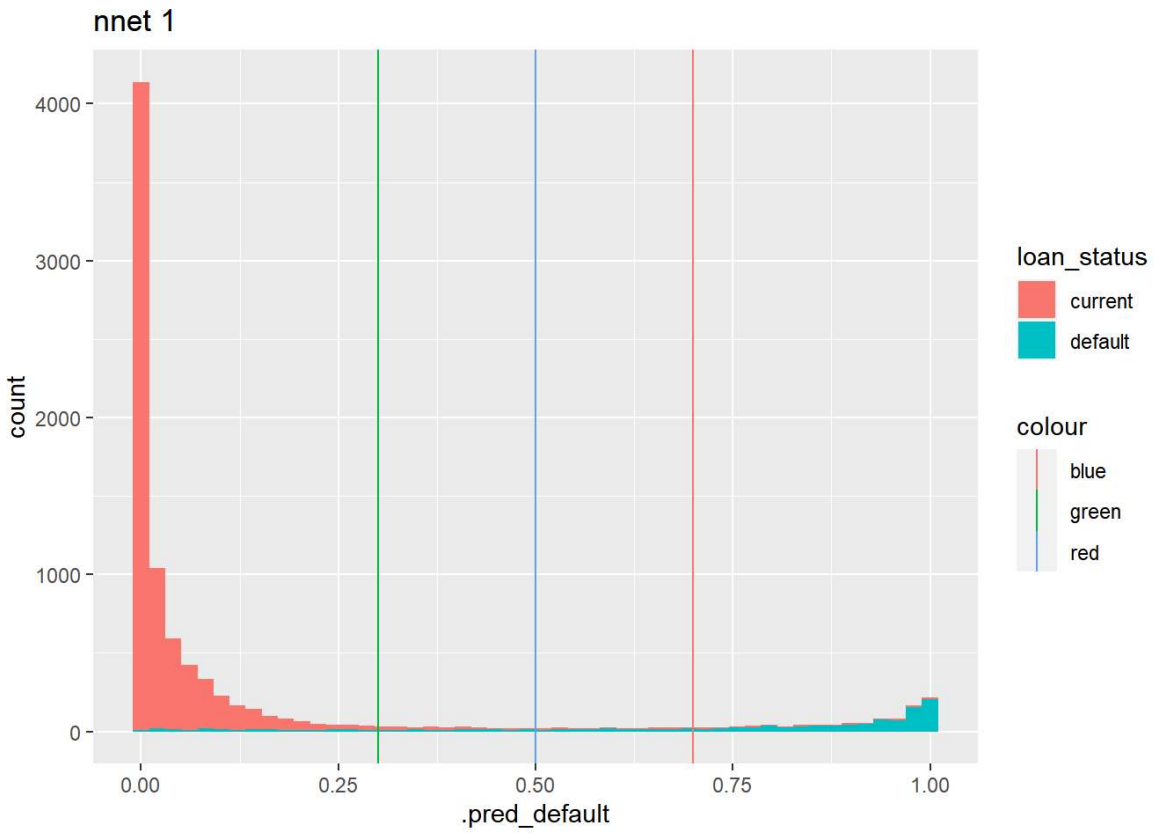
```
logistic_train_mutate <- logistic_train %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current')))

logistic_test_mutate <- logistic_test %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current')))

bind_rows(logistic_train, logistic_test) %>%
    group_by(part) %>%
    dplyr::select(loan_status, .pred_class) -> train_test

precision <- train_test %>%
  yardstick::precision(loan_status, .pred_class)

recall <- train_test %>%
  yardstick::recall(loan_status, .pred_class)

logistic_train %>%
    metrics(loan_status, .pred_default, estimate = .pred_class) %>%
    bind_rows(logistic_train_mutate %>% yardstick::precision(loan_status, predict_class)) %>%
    bind_rows(logistic_train_mutate %>% yardstick::recall(loan_status, predict_class)) %>%
    filter(.metric %in% c("accuracy", "roc_auc", "mn_log_loss", "precision", "recall")) %>%
    mutate(part="training") %>%
bind_rows(logistic_test %>%
    metrics(loan_status, .pred_default, estimate = .pred_class) %>%
    bind_rows(logistic_test_mutate %>% yardstick::precision(loan_status, predict_class)) %>%
    bind_rows(logistic_test_mutate %>% yardstick::recall(loan_status, predict_class)) %>%
    filter(.metric %in% c("accuracy", "roc_auc", "mn_log_loss", "precision", "recall")) %>%
    mutate(part="testing"))  %>%
bind_rows(
  bind_cols(precision, recall) %>%
    mutate(f1_score = 2*.estimate...4*.estimate...8/(.estimate...4 + .estimate...8)) %>%
    mutate(.metric = "F1_Score") %>%
    mutate(.estimator = "binary") %>%
    dplyr::select(part...1, .metric, .estimator, f1_score) %>%
    rename(c(".estimate" = "f1_score"))
) %>%
    arrange(desc(part))
```

.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
accuracy	binary	0.9438100	training
mn_log_loss	binary	0.1525445	training
roc_auc	binary	0.9695320	training
precision	binary	0.8641661	training
recall	binary	0.7483444	training
F1_Score	binary	0.8020957	training
accuracy	binary	0.9458128	testing

.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
mn_log_loss	binary	0.1537686	testing
roc_auc	binary	0.9682043	testing
precision	binary	0.8462838	testing

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

```
# Score Distribution
logistic_test %>%
  ggplot(aes(.pred_default, fill=loan_status)) +
  geom_histogram(bins=50) +
  geom_vline(aes(xintercept=.5, color="red")) +
  geom_vline(aes(xintercept=.3, color="green")) +
  geom_vline(aes(xintercept=.7, color="blue")) +
  labs(title = "nnet 1") -> scored_dist

print(scored_dist)
```



```
# operating range 0 - 10%
operating_range <- logistic_test %>%
  roc_curve(loan_status, .pred_default)  %>%
  mutate(
    fpr = round((1 - specificity), 3),
    tpr = round(sensitivity, 3),
    score_threshold =  round(.threshold, 5)
  ) %>%
  group_by(fpr) %>%
  summarise(threshold = round(mean(score_threshold),4),
            tpr = mean(tpr)) %>%
  filter(fpr <= 0.5)

  print(operating_range)
```

```
## # A tibble: 501 × 3
##      fpr threshold    tpr
##    <dbl>     <dbl>  <dbl>
##  1 0       Inf     0.0547
##  2 0.001     0.991 0.142
##  3 0.002     0.981 0.219
##  4 0.003     0.965 0.289
##  5 0.004     0.947 0.335
##  6 0.005     0.929 0.384
##  7 0.006     0.905 0.429
##  8 0.007     0.886 0.459
##  9 0.008     0.862 0.496
## 10 0.009     0.840 0.522
## # … with 491 more rows
```

```
# Precision Recall Chart
logistic_test %>%
  pr_curve(loan_status, .pred_default) %>%
  mutate(
    recall = round(recall, 2),
    .threshold = round(.threshold, 3),
    precision = round(precision, 3)
  ) %>%
  group_by(recall) %>%
  summarise(precision = max(precision),
            .threshold = min(.threshold))
```

recall	precision	.threshold
<dbl>	<dbl>	<dbl>
0.00	1.000	1.000
0.01	1.000	0.999
0.02	1.000	0.999
0.03	1.000	0.998

recall	precision	.threshold
<dbl>	<dbl>	<dbl>
0.04	0.983	0.998
0.05	0.986	0.997
0.06	0.988	0.997
0.07	0.990	0.996
0.08	0.990	0.995
0.09	0.983	0.994

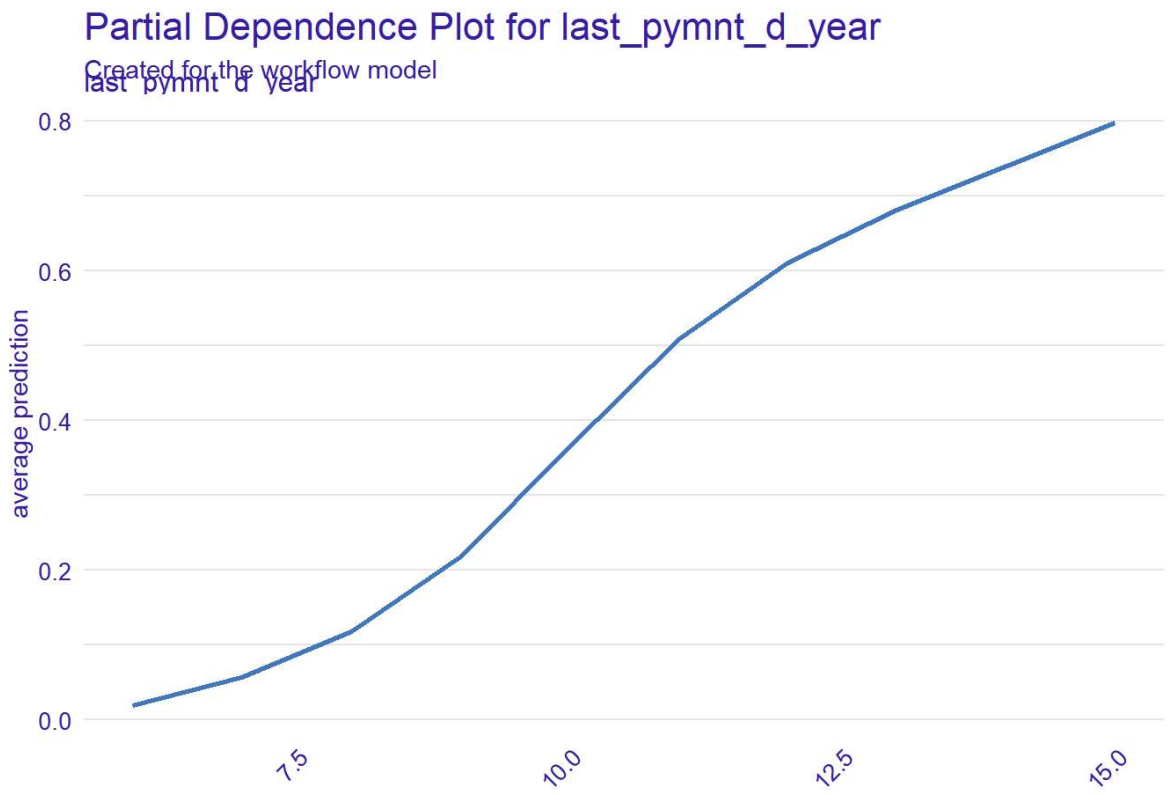
1-10 of 101 rows Previous 1 2 3 4 5 6 ... 11 Next

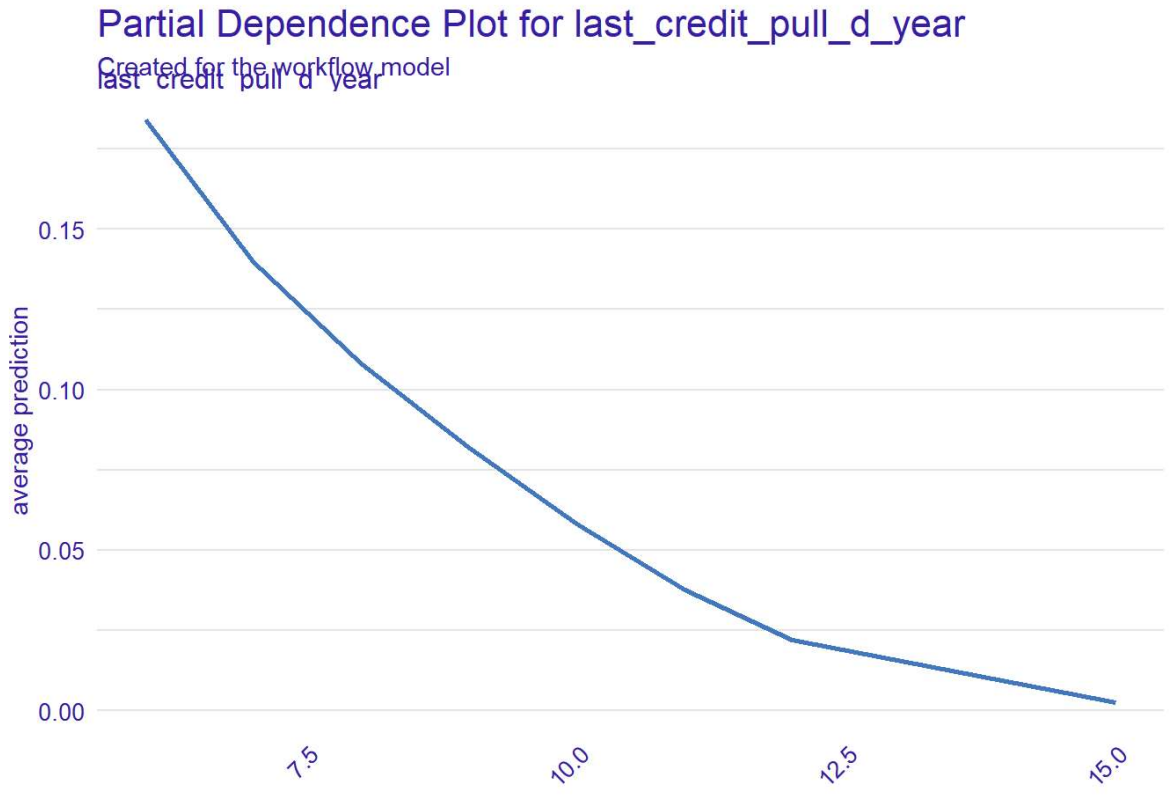
Partial Dependance Plot (Logistic Regression - Full Model)

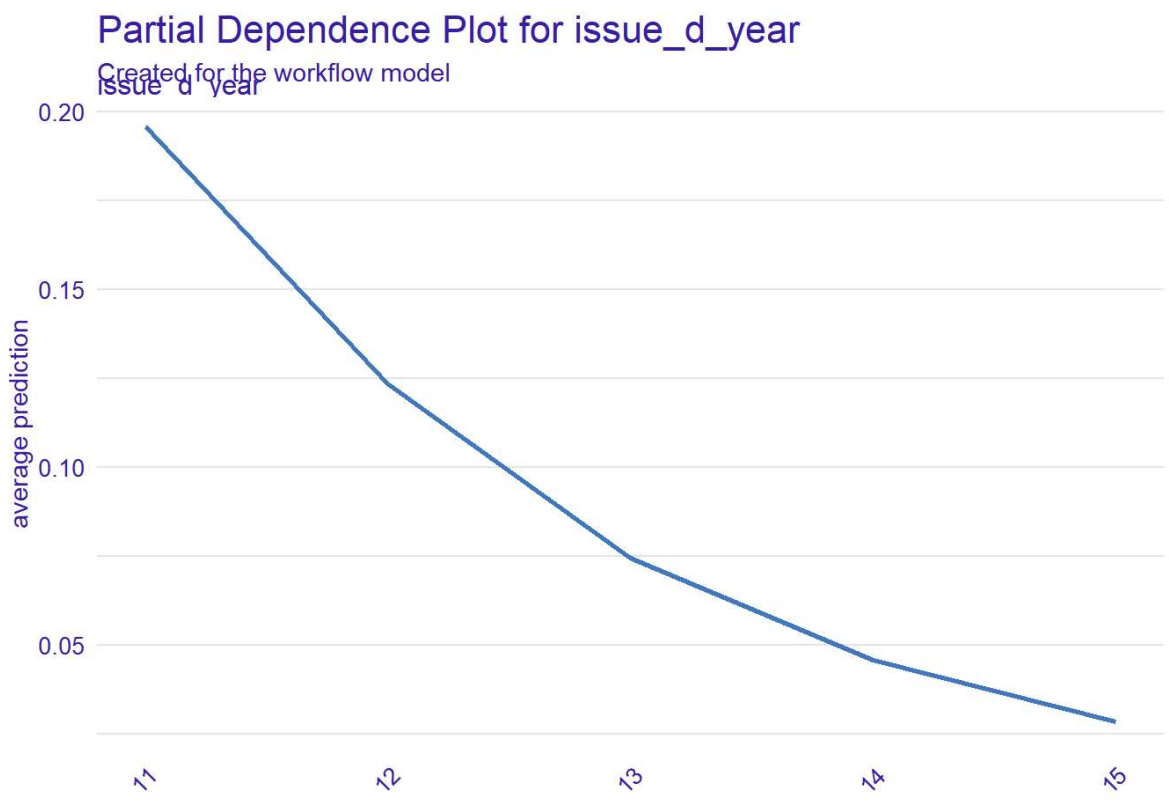
```
# create an explainer of a model


logistic_explainer <- explain_tidymodels(
  logistic_wf,
  data = train ,
  y = train$loan_default ,
  verbose = TRUE
)
```

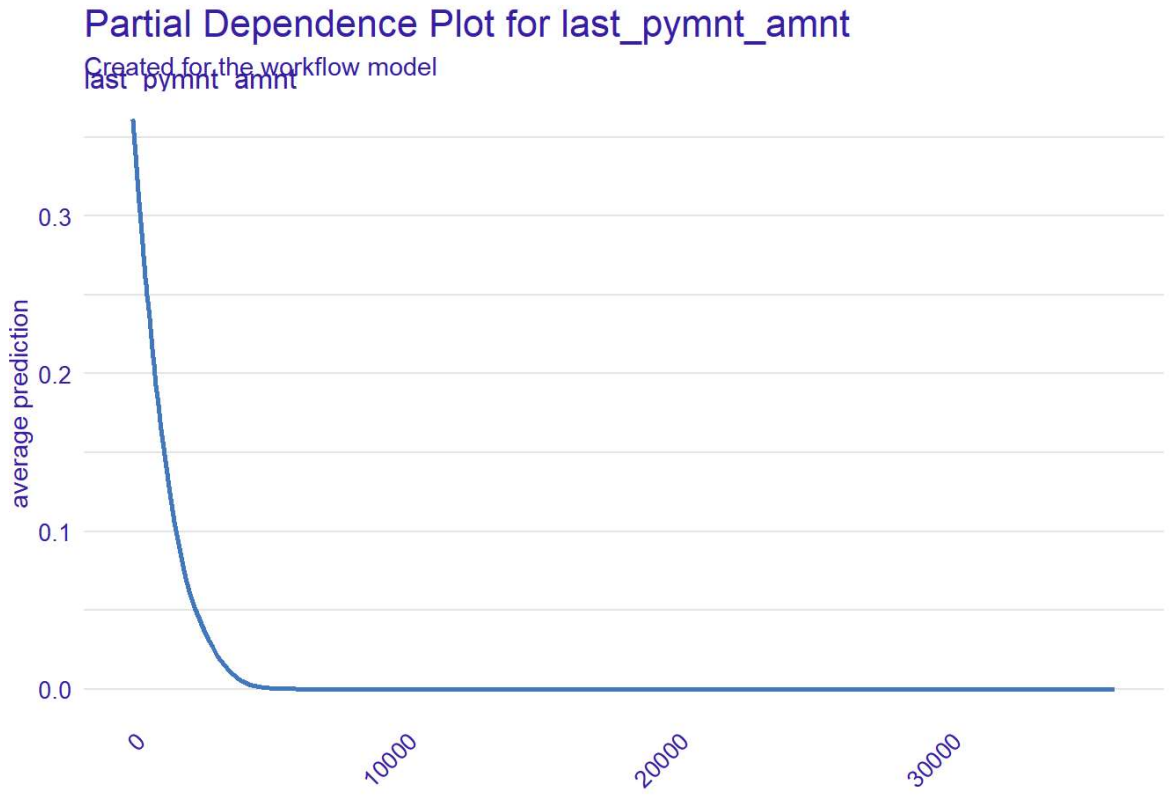
```
## Preparation of a new explainer is initiated
##   -> model label       :  workflow  (  default  )
##   -> data              :  20840  rows  32  cols
##   -> data              :  tibble converted into a data.frame
##   -> target variable   :  not specified! (  WARNING  )
##   -> predict function  :  yhat.workflow  will be used (  default  )
##   -> predicted values  :  No value for predict function target column. (  default  )
##   -> model_info        :  package tidymodels , ver. 1.0.0 , task classification (  default  )
##   -> model_info        :  Model info detected classification task but 'y' is a NULL .  (  WARNING  )
##   -> model_info        :  By deafult classification tasks supports only numercical 'y' parameter.
##   -> model_info        :  Consider changing to numerical vector with 0 and 1 values.
##   -> model_info        :  Otherwise I will not be able to calculate residuals or loss function.
##   -> predicted values  :  numerical, min =  0.00000000000000022204 , mean =  0.1518757 , max =  0.9999777
##   -> residual function :  difference between y and yhat (  default  )
##   A new explainer has been created!
```

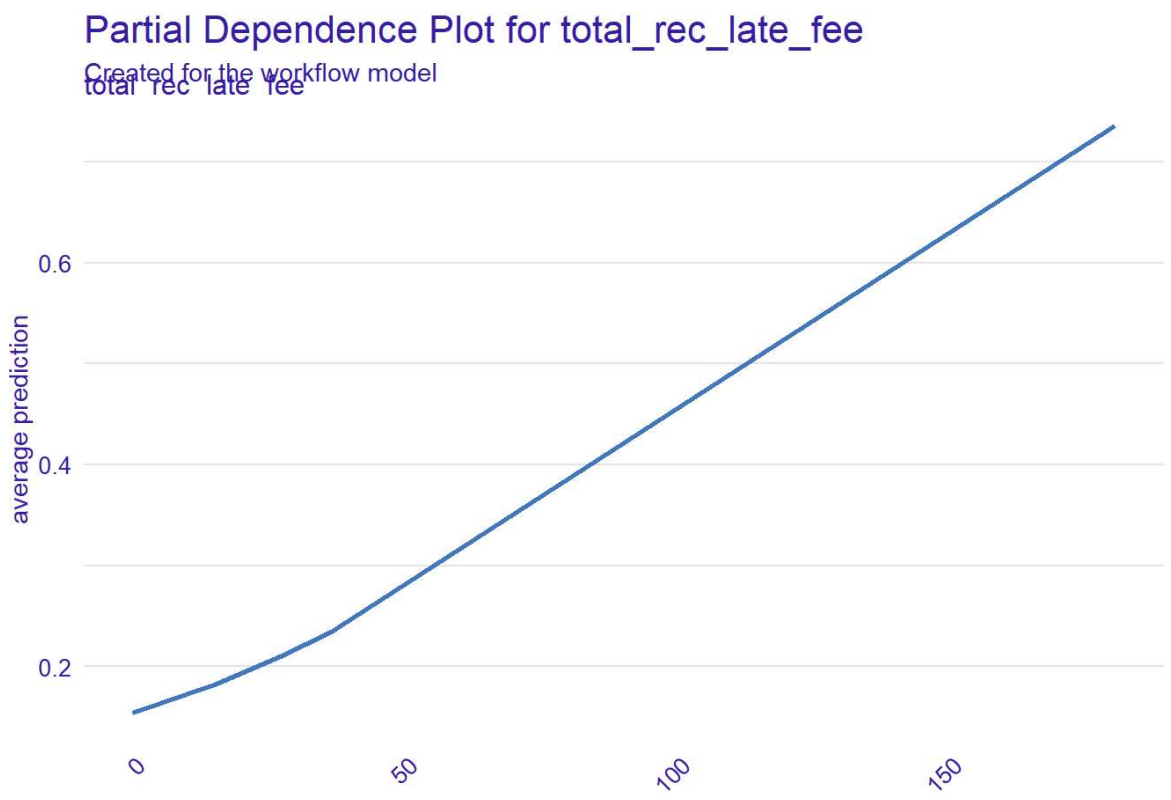
```
logistic_variable <- c('last_pymnt_d_year', 'last_credit_pull_d_year', 'issue_d_year', 'last_pymnt_amnt', 'total_rec_late_fe
         e')


pdp <- function(m){

# create a profile of a single variable for a model

pdp_variable <- model_profile(
  logistic_explainer,
  variables = as.character(m)
)

# Plot it

plot(pdp_variable) +
  ggtitle(paste0("Partial Dependence Plot for ", as.character(m))) +
  theme(axis.text.x=element_text(angle=45, hjust=1))

}

for (c in logistic_variable){
    print(pdp(c))
}
```


Partial Dependence Plot for last_pymnt_d_year
Created for the workflow model
last_pymnt_d_year
average prediction
0.8
0.6
0.4
0.2
0.0
7.5
10.0
12.5
15.0


Partial Dependence Plot for last_credit_pull_d_year
Created for the workflow model
last_credit_pull_d_year
average prediction
0.15
0.10
0.05
0.00
7.5
10.0
12.5
15.0


Partial Dependence Plot for issue_d_year
Created for the workflow model
issue_d_year
0.20
0.15
0.10
0.05
average prediction
11
12
13
14
15


Partial Dependence Plot for last_pymnt_amnt
Created for the workflow model
last_pymnt_amnt
0.3
0.2
0.1
0.0
average prediction
0
10000
20000
30000



Lasso

Lasso L1 regularization

Here we use the hyper parameters

```
lasso_spec <- logistic_reg(penalty = 0.01, mixture = 1) %>%
  set_mode("classification") %>%
  set_engine("glmnet")

lasso_wf <- workflow() %>%
  add_recipe(loan_recipe) %>%
  add_model(lasso_spec) %>%
  fit(train)

lasso_wf %>%
 pull_workflow_fit() %>%
  tidy() %>%
  filter(estimate != 0)
```

term	estimate	penalty
<chr>	<dbl>	<dbl>
(Intercept)	-2.08913703	0.01
loan_amnt	0.12717119	0.01
int_rate	0.27842407	0.01
total_rec_late_fee	0.18804891	0.01

term	estimate	penalty
<chr>	<dbl>	<dbl>
last_pymnt_amnt	-2.44474992	0.01
issue_d_year	-0.68560973	0.01
last_pymnt_d_year	1.67780666	0.01
last_credit_pull_d_year	-1.03085043	0.01
term_X36.months	-1.22695142	0.01
term_X60.months	0.32174797	0.01

1-10 of 12 rows Previous 1 2 Next

```
predict(lasso_wf, train, type="prob") %>%
  bind_cols(predict(logistic_wf, train, type="class")) %>%
  bind_cols(train) -> lasso_train

predict(lasso_wf, test, type="prob") %>%
  bind_cols(predict(lasso_wf, test, type="class")) %>%
  bind_cols(test) -> lasso_test

lasso_train %>%
  metrics(loan_status, estimate = .pred_class, .pred_default) %>%
  mutate(part="training") %>%
bind_rows(lasso_test %>%
  metrics(loan_status, estimate = .pred_class, .pred_default) %>%
  mutate(part="testing"))
```

.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
accuracy	binary	0.9438100	training
kap	binary	0.7695490	training
mn_log_loss	binary	0.1922341	training
roc_auc	binary	0.9582208	training
accuracy	binary	0.9312584	testing
kap	binary	0.6869007	testing
mn_log_loss	binary	0.1853371	testing
roc_auc	binary	0.9615854	testing

8 rows

```
# variables in recipe: loan_amnt, annual_inc, inq_last_6mths, last_pymnt_amnt, term, int_rate, emp_length, purpose, issue_d_y
         ear, last_pymnt_year, grade, last_credit_pull_year, funded_amnt, fico_range_low, total_rec_late_fee
```

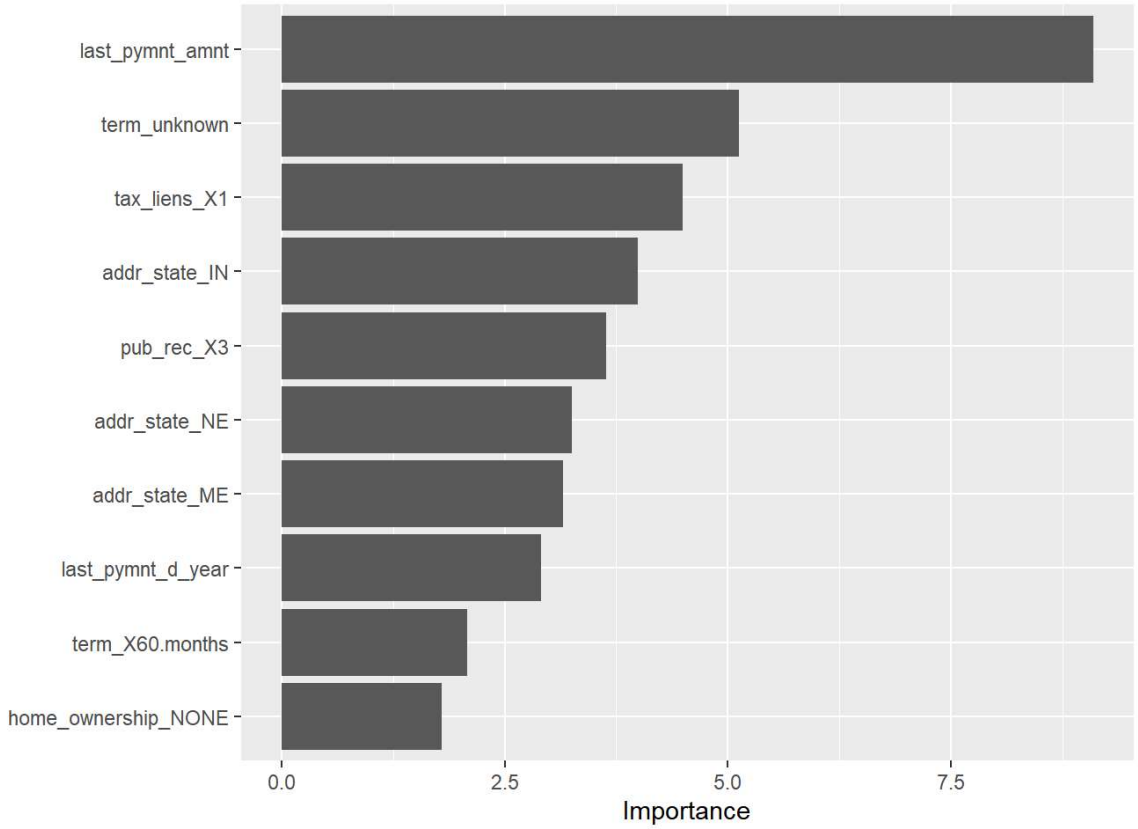
Lasso Evaluation

```
options(yardstick.event_first = FALSE)

# Variable Importance
lasso_wf %>%
  extract_fit_parsnip() %>%
  vi()
```

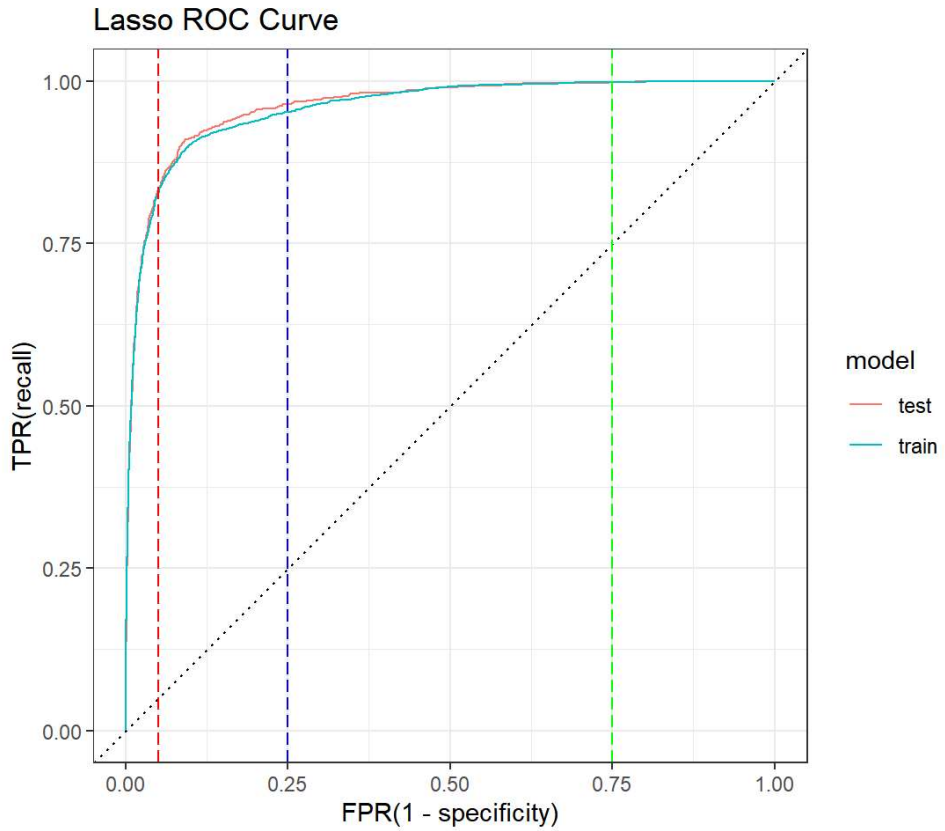
Variable	Importance	Sign
<chr>	<dbl>	<chr>
last_pymnt_amnt	9.0923261129567833904730	NEG
term_unknown	5.1286336458835339868756	NEG
tax_liens_X1	4.4943443497435566769127	NEG
addr_state_IN	3.9946258234642399997938	POS
pub_rec_X3	3.6388044429070309782048	NEG
addr_state_NE	3.2527611714423994015988	POS
addr_state_ME	3.1520932394932330566917	NEG
last_pymnt_d_year	2.9040030398671792255527	POS
term_X60.months	2.0822768611670832150651	POS
home_ownership_NONE	1.7967065953353615892496	POS

1-10 of 177 rows Previous **1** 2 3 4 5 6 ... 18 Next

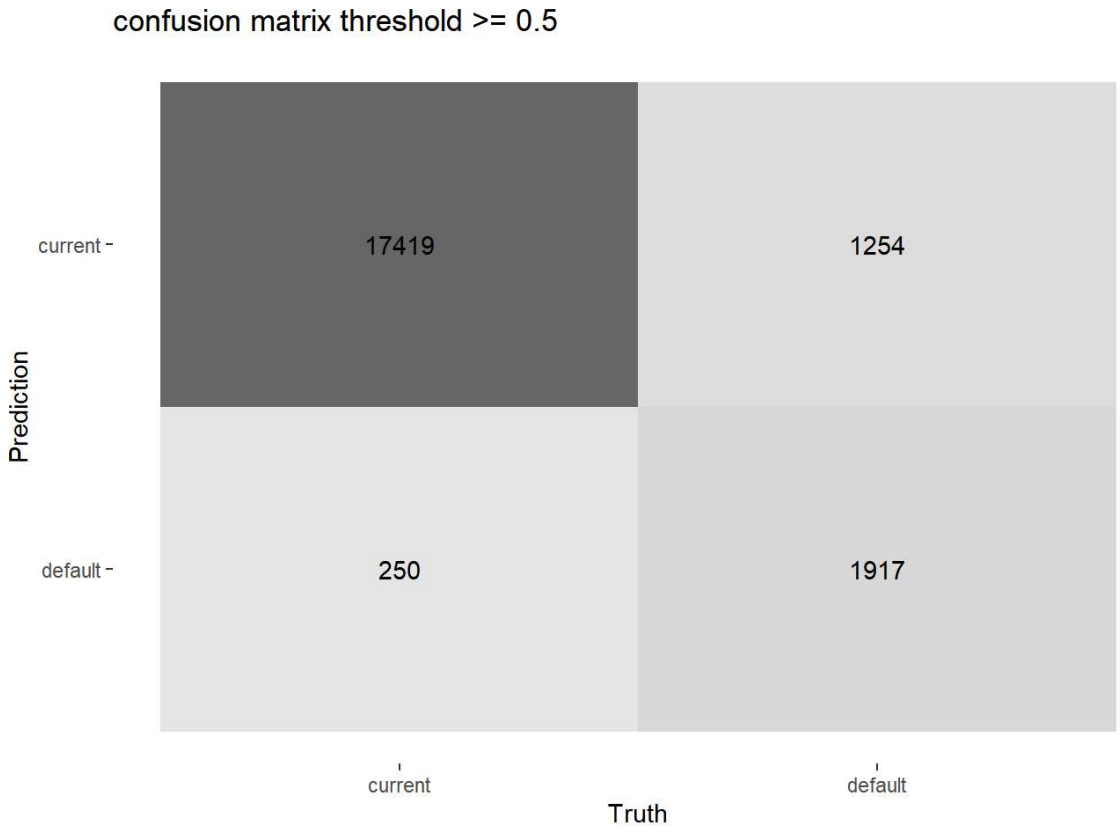
```
lasso_wf %>%
  extract_fit_parsnip() %>%
  vip()
```



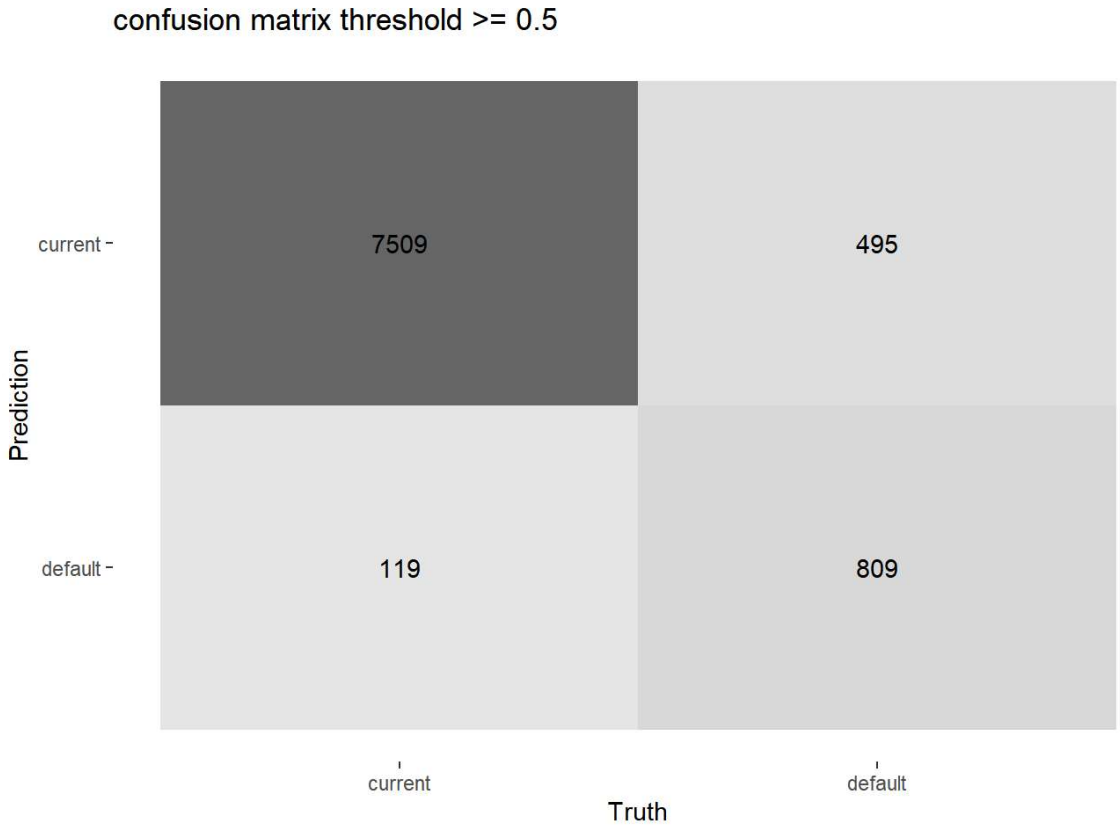

```
lasso_train <- lasso_train %>% mutate(part = "training")

lasso_test <- lasso_test %>% mutate(part = "testing")

lasso_train %>% mutate(model = "train") %>%
  bind_rows(lasso_test %>% mutate(model="test")) %>%
  group_by(model) %>%
  roc_curve(loan_status, .pred_default) %>%
  autoplot() +
  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 = "Lasso ROC Curve" , x = "FPR(1 - specificity)", y = "TPR(recall)")
```




```
lasso_train %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current'))) %>%
   conf_mat(loan_status, estimate = predict_class) %>%
   autoplot(type = "heatmap") +
   labs(title="confusion matrix threshold >= 0.5")
```



```
lasso_test %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current'))) %>%
   conf_mat(loan_status, estimate = predict_class) %>%
   autoplot(type = "heatmap") +
   labs(title="confusion matrix threshold >= 0.5")
```



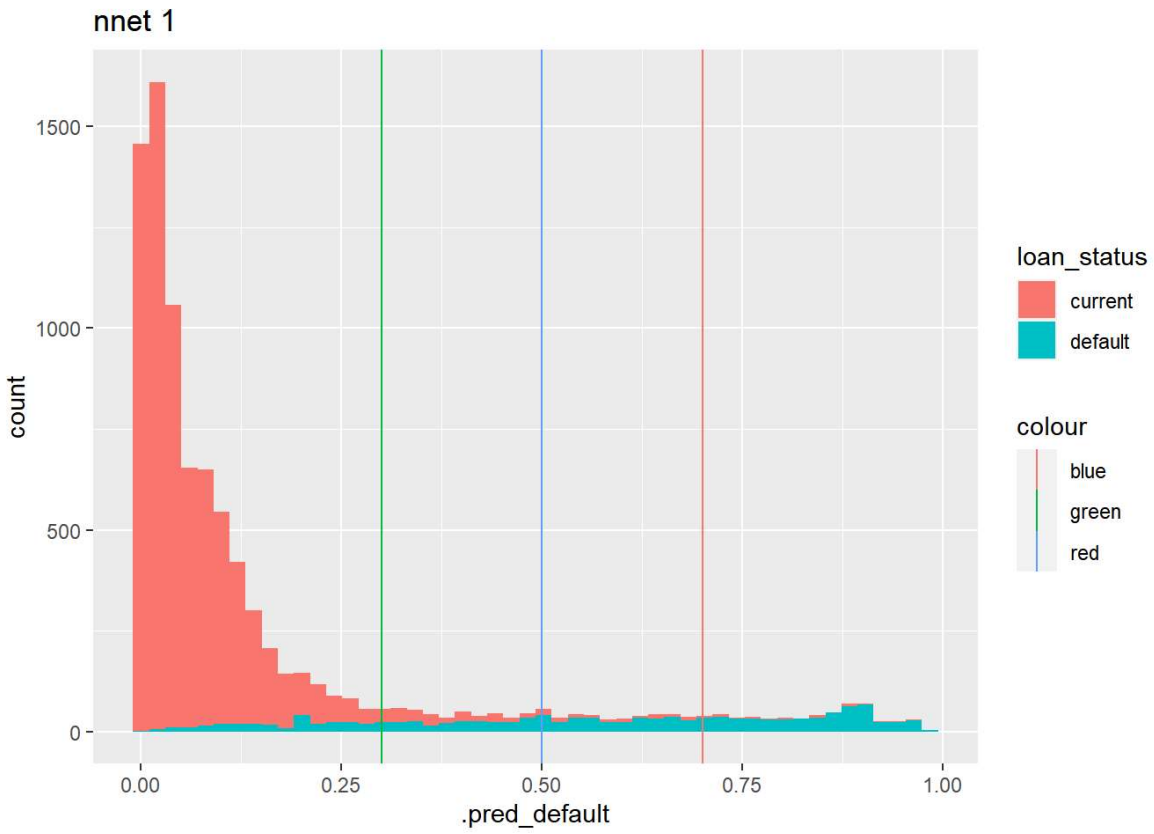
```
lasso_train_mutate <- lasso_train %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current')))

lasso_test_mutate <- lasso_test %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current')))

bind_rows(lasso_train, lasso_test) %>%
    group_by(part) %>%
    dplyr::select(loan_status, .pred_class) -> train_test

precision <- train_test %>%
  yardstick::precision(loan_status, .pred_class)

recall <- train_test %>%
  yardstick::recall(loan_status, .pred_class)

lasso_train %>%
    metrics(loan_status, .pred_default, estimate = .pred_class) %>%
    bind_rows(logistic_train_mutate %>% yardstick::precision(loan_status, predict_class)) %>%
    bind_rows(logistic_train_mutate %>% yardstick::recall(loan_status, predict_class)) %>%
    filter(.metric %in% c("accuracy", "roc_auc", "mn_log_loss", "precision", "recall")) %>%
    mutate(part="training") %>%
bind_rows(lasso_test %>%
    metrics(loan_status, .pred_default, estimate = .pred_class) %>%
    bind_rows(logistic_test_mutate %>% yardstick::precision(loan_status, predict_class)) %>%
    bind_rows(logistic_test_mutate %>% yardstick::recall(loan_status, predict_class)) %>%
    filter(.metric %in% c("accuracy", "roc_auc", "mn_log_loss", "precision", "recall")) %>%
    mutate(part="testing"))  %>%
bind_rows(
  bind_cols(precision, recall) %>%
    mutate(f1_score = 2*.estimate...4*.estimate...8/(.estimate...4 + .estimate...8)) %>%
    mutate(.metric = "F1_Score") %>%
    mutate(.estimator = "binary") %>%
    dplyr::select(part...1, .metric, .estimator, f1_score) %>%
    rename(c(".estimate" = "f1_score"))
) %>%
    arrange(desc(part))
```

.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
accuracy	binary	0.9438100	training
mn_log_loss	binary	0.1922341	training
roc_auc	binary	0.9582208	training
precision	binary	0.8641661	training
recall	binary	0.7483444	training
F1_Score	binary	0.8020957	training
accuracy	binary	0.9312584	testing

.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
mn_log_loss	binary	0.1853371	testing
roc_auc	binary	0.9615854	testing
precision	binary	0.8462838	testing

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

```
# Score Distribution
lasso_test %>%
  ggplot(aes(.pred_default, fill=loan_status)) +
  geom_histogram(bins=50) +
  geom_vline(aes(xintercept=.5, color="red")) +
  geom_vline(aes(xintercept=.3, color="green")) +
  geom_vline(aes(xintercept=.7, color="blue")) +
  labs(title = "nnet 1") -> scored_dist

print(scored_dist)
```



```
# operating range 0 - 10%
operating_range <- lasso_test %>%
  roc_curve(loan_status, .pred_default)  %>%
  mutate(
    fpr = round((1 - specificity), 3),
    tpr = round(sensitivity, 3),
    score_threshold =  round(.threshold, 5)
  ) %>%
  group_by(fpr) %>%
  summarise(threshold = round(mean(score_threshold),4),
            tpr = mean(tpr)) %>%
  filter(fpr <= 0.5)

  print(operating_range)
```

```
## # A tibble: 501 × 3
##      fpr threshold    tpr
##    <dbl>     <dbl>  <dbl>
##  1 0       Inf     0.0408
##  2 0.001     0.881 0.140
##  3 0.002     0.829 0.228
##  4 0.003     0.783 0.281
##  5 0.004     0.737 0.338
##  6 0.005     0.700 0.387
##  7 0.006     0.674 0.415
##  8 0.007     0.655 0.443
##  9 0.008     0.634 0.468
## 10 0.009     0.613 0.495
## # … with 491 more rows
```

```
# Precision Recall Chart
lasso_test %>%
  pr_curve(loan_status, .pred_default) %>%
  mutate(
    recall = round(recall, 2),
    .threshold = round(.threshold, 3),
    precision = round(precision, 3)
  ) %>%
  group_by(recall) %>%
  summarise(precision = max(precision),
            .threshold = min(.threshold))
```

recall	precision	.threshold
<dbl>	<dbl>	<dbl>
0.00	1.000	0.971
0.01	1.000	0.960
0.02	0.970	0.953
0.03	0.978	0.946

recall	precision	.threshold
<dbl>	<dbl>	<dbl>
0.04	0.967	0.932
0.05	0.970	0.919
0.06	0.966	0.912
0.07	0.970	0.907
0.08	0.973	0.904
0.09	0.969	0.902

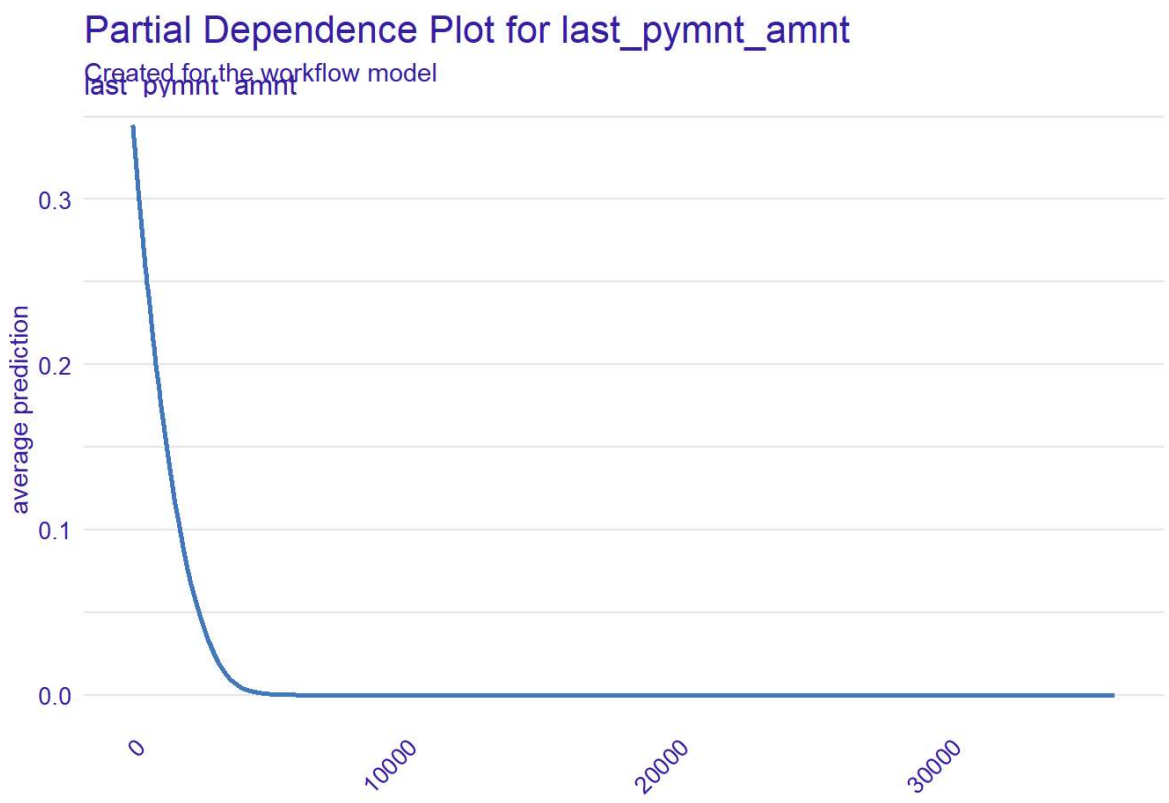
1-10 of 101 rows Previous 1 2 3 4 5 6 ... 11 Next

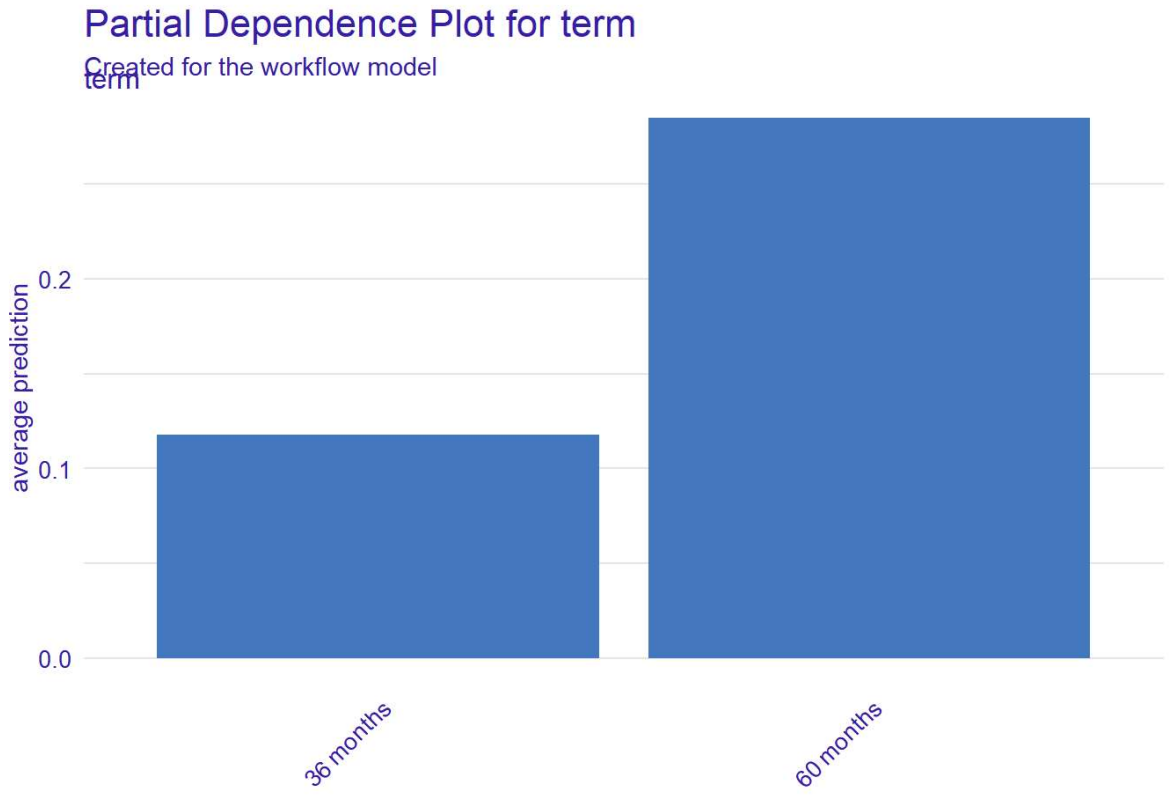
Partial Dependance Plot (Lasso)

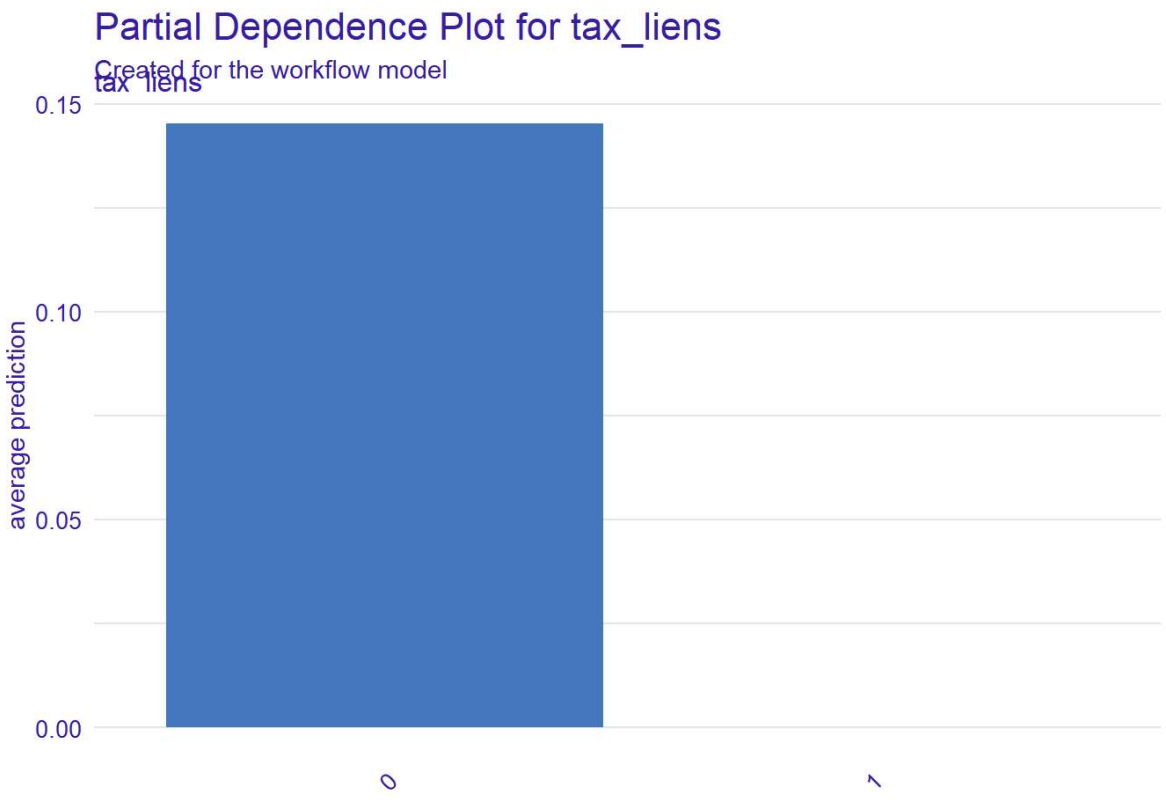
```
# create an explainer of a model


lasso_explainer <- explain_tidymodels(
  logistic_wf,
  data = train ,
  y = train$loan_default ,
  verbose = TRUE
)
```

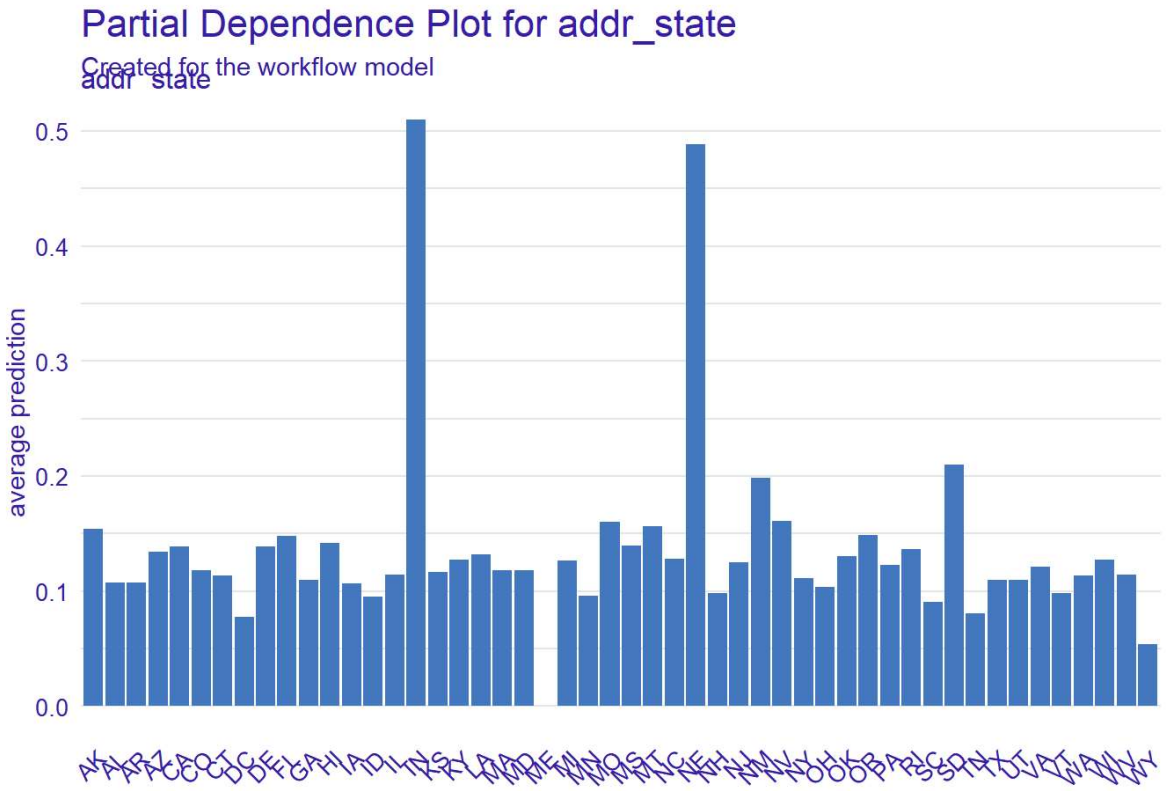
```
## Preparation of a new explainer is initiated
##   -> model label       :  workflow  (  default  )
##   -> data              :  20840  rows  32  cols
##   -> data              :  tibble converted into a data.frame
##   -> target variable   :  not specified! (  WARNING  )
##   -> predict function  :  yhat.workflow  will be used (  default  )
##   -> predicted values  :  No value for predict function target column. (  default  )
##   -> model_info        :  package tidymodels , ver. 1.0.0 , task classification (  default  )
##   -> model_info        :  Model info detected classification task but 'y' is a NULL .  (  WARNING  )
##   -> model_info        :  By deafult classification tasks supports only numercical 'y' parameter.
##   -> model_info        :  Consider changing to numerical vector with 0 and 1 values.
##   -> model_info        :  Otherwise I will not be able to calculate residuals or loss function.
##   -> predicted values  :  numerical, min =  0.00000000000000002220446 , mean =  0.1518757 , max =  0.9999777
##   -> residual function :  difference between y and yhat (  default  )
##   A new explainer has been created!
```

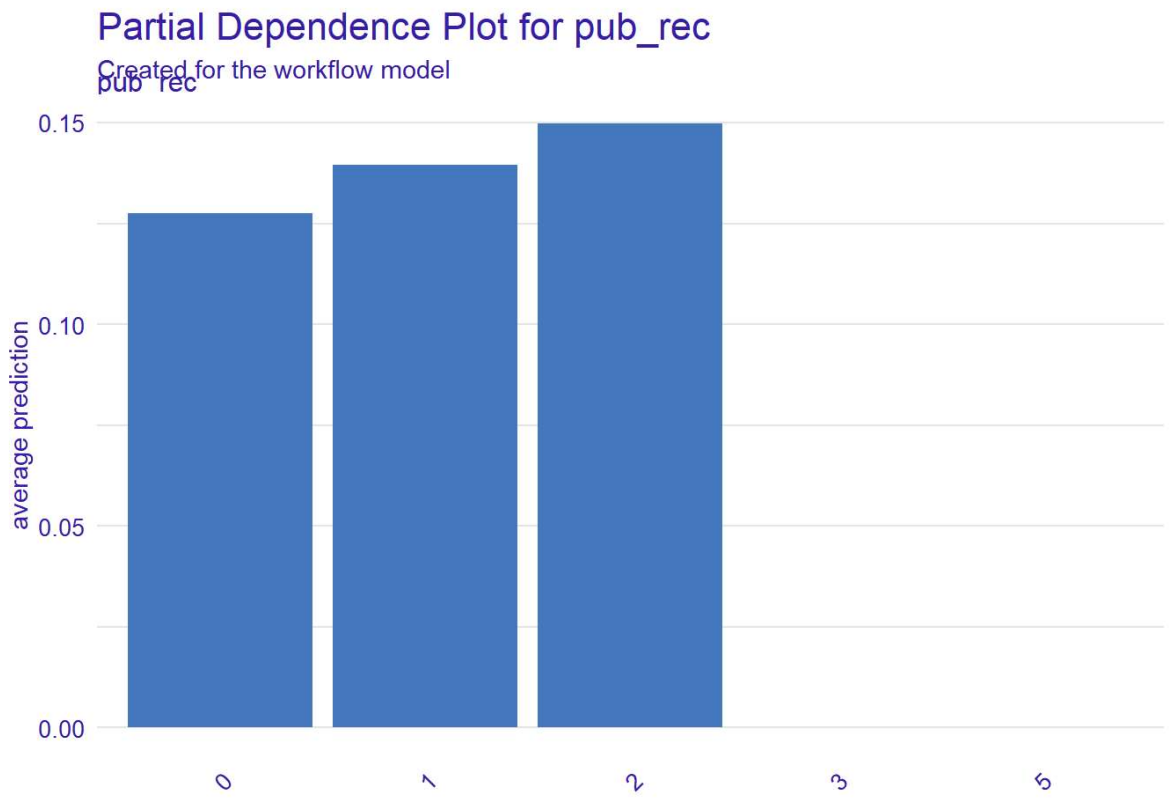
```
lasso_variable <- c('last_pymnt_amnt', 'term', 'tax_liens', 'addr_state', 'pub_rec')

pdp <- function(m){

# create a profile of a single variable for a model

pdp_variable <- model_profile(
  lasso_explainer,
  variables = as.character(m)
)

# Plot it

plot(pdp_variable) +
  ggtitle(paste0("Partial Dependence Plot for ", as.character(m))) +
  theme(axis.text.x=element_text(angle=45, hjust=1))

}

for (c in lasso_variable){
    print(pdp(c))
}
```


Partial Dependence Plot for last_pymnt_amnt
Created for the workflow model
last_pymnt_amnt
average prediction
0.3
0.2
0.1
0.0
0
10000
20000
30000


Partial Dependence Plot for term
Created for the workflow model
term
average prediction
0.2
0.1
0.0
36 months
60 months


Partial Dependence Plot for tax_liens
Created for the workflow model
tax_liens
average prediction
0.15
0.10
0.05
0.00
0
1


Partial Dependence Plot for addr_state
Created for the workflow model
addr_state
average prediction
0.5
0.4
0.3
0.2
0.1
0.0
AK AL AR AZ CA CO CT DC DE FL GA HI IA ID IL IN KS KY LA MA MD ME MI MN MO MS MT NC NE NH NJ NM NV NY OH OK OR PA RI SC SD TN TX UT VA VT WA WI WV WY



Define Recipe & Bake

```
recipe <- recipe(loan_status ~ loan_amnt + annual_inc + inq_last_6mths + last_pymnt_amnt + term + int_rate + emp_length + pur
          pose + issue_d_year + last_pymnt_d_year + grade + last_credit_pull_d_year + funded_amnt + fico_range_low + total_re
          c_late_fee, data=train) %>%
    step_impute_median(all_numeric_predictors()) %>%
    step_unknown(all_nominal_predictors()) %>%
    step_scale(all_numeric_predictors()) %>%
    step_novel(all_nominal_predictors()) %>% # new factor levels
    step_dummy(all_nominal_predictors(), one_hot = TRUE) %>%
    step_nzv(all_predictors()) %>%
    prep()


recipe
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor         15
##
## Training data contained 20840 data points and 60 incomplete rows.
##
## Operations:
##
## Median imputation for loan_amnt, annual_inc, inq_last_6mths, last_pym... [trained]
## Unknown factor level assignment for term, emp_length, purpose, grade [trained]
## Scaling for loan_amnt, annual_inc, inq_last_6mths, last_pym... [trained]
## Novel factor level assignment for term, emp_length, purpose, grade [trained]
## Dummy variables from term, emp_length, purpose, grade [trained]
## Sparse, unbalanced variable filter removed total_rec_late_fee, term_unknown, term_... [trained]
```

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

loan_amnt <dbl>	annual_inc <dbl>	inq_last_6mths <dbl>	last_pymnt_amnt <dbl>	int_rate <dbl>	issue_d_year <dbl>	last_pymnt_d_year <dbl>
2.6547687	1.3465008	0.000000	0.116002320	4.846935	11.39485	4.952620
0.2979842	0.9350700	0.660703	0.018050400	3.499816	11.39485	4.952620
0.7348021	1.4213064	1.982109	0.046227487	2.287409	11.39485	4.952620
2.7089477	0.7854588	0.660703	2.332716406	4.103325	11.39485	4.952620
0.6772369	1.1220840	3.303515	0.007674276	2.314351	13.46664	6.190775
1.5576449	1.0472784	0.000000	0.920458509	1.883273	11.39485	5.571697

6 rows | 1-7 of 31 columns

```
bake_train <- bake(recipe, new_data = train)
bake_test  <- bake(recipe, new_data = test)
```

Neural Network

Define Neural Network Model

```
# K-fold cross validation
kfold_splits <- vfold_cv(train, v=5)

nn_model <- mlp(hidden_units = tune(),
                penalty=tune(),
  epochs = tune(),
  ) %>%
  set_engine("nnet") %>%
  set_mode("classification")

nn_wflow <-workflow() %>%
  add_recipe(recipe) %>%
  add_model(nn_model)

nn_search_res <- nn_wflow %>%
  tune_bayes(
    resamples = kfold_splits,
    # Generate five at semi-random to start
    initial = 5,
    iter = 50,
    # How to measure performance?
    metrics = metric_set(yardstick::roc_auc),
    control = control_bayes(no_improve = 5, verbose = TRUE)
  )
```

```
##
```

```
## ❯ Generating a set of 5 initial parameter results
```

```
## ✓ Initialization complete
```

```
##
```

```
##
```

```
## ──── Iteration 1 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9752 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i hidden_units=10, penalty=0.00713, epochs=13
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.8772 (+/-0.0232)
```

```
##
```

```
## ── Iteration 2 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9752 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i hidden_units=8, penalty=0.000000429, epochs=379
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.9642 (+/-0.00384)
```

```
##
```

```
## ─── Iteration 3 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9752 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i hidden_units=7, penalty=0.819, epochs=171
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.9743 (+/-0.000951)
```

```
##
```

```
## ─── Iteration 4 ──────────────────────────────────────────────────────────────────────
──────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9752 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i hidden_units=10, penalty=0.125, epochs=658
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ♥ Newest results:    roc_auc=0.9758 (+/-0.00147)
```

```
##
```

```
## ── Iteration 5 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9758 (@iter 4)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i hidden_units=6, penalty=0.00000000103, epochs=12
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.8025 (+/-0.052)
```

```
##
```

```
## ── Iteration 6 ──────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9758 (@iter 4)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i hidden_units=3, penalty=0.00000000757, epochs=170
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.8459 (+/-0.101)
```

```
##
```

```
## ─── Iteration 7 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9758 (@iter 4)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i hidden_units=9, penalty=0.000000188, epochs=582
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.9659 (+/-0.00371)
```

```
##
```

```
## ─── Iteration 8 ──────────────────────────────────────────────────────────────────────────────
──────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9758 (@iter 4)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i hidden_units=8, penalty=0.000000000154, epochs=253
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.969 (+/-0.00238)
```

```
##
```

```
## ── Iteration 9 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9758 (@iter 4)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i hidden_units=7, penalty=0.00000000133, epochs=239
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.9665 (+/-0.00453)
```

```
## ! No improvement for 5 iterations; returning current results.
```

NNET Tuning

Evaluate our tuning efforts

```
# Experiments
nn_search_res %>%
  collect_metrics()
```

hidden_units	penalty	epo...	.metric	.estimator	mean	n	std_err	.config
<int>	<dbl>	<int>	<chr>	<chr>	<dbl>	<int>	<dbl>	<chr>
4	0.000000064280157	266	roc_auc	binary	0.8113193	5	0.0962058273	Preprocessor1_Model1
9	0.143887610428777	728	roc_auc	binary	0.9752137	5	0.0014778317	Preprocessor1_Model2
6	0.000000003752284	165	roc_auc	binary	0.9702852	5	0.0024799427	Preprocessor1_Model3
2	0.000423997562159	569	roc_auc	binary	0.5881283	5	0.0921970202	Preprocessor1_Model4
8	0.000001197843786	919	roc_auc	binary	0.8687785	5	0.1025408523	Preprocessor1_Model5
10	0.007130001815917	13	roc_auc	binary	0.8771867	5	0.0232379648	Iter1
8	0.000000429213322	379	roc_auc	binary	0.9642297	5	0.0038422555	Iter2
7	0.819164178460531	171	roc_auc	binary	0.9743222	5	0.0009510775	Iter3
10	0.125119871896481	658	roc_auc	binary	0.9757630	5	0.0014740683	Iter4
6	0.000000001028791	12	roc_auc	binary	0.8024517	5	0.0520170551	Iter5

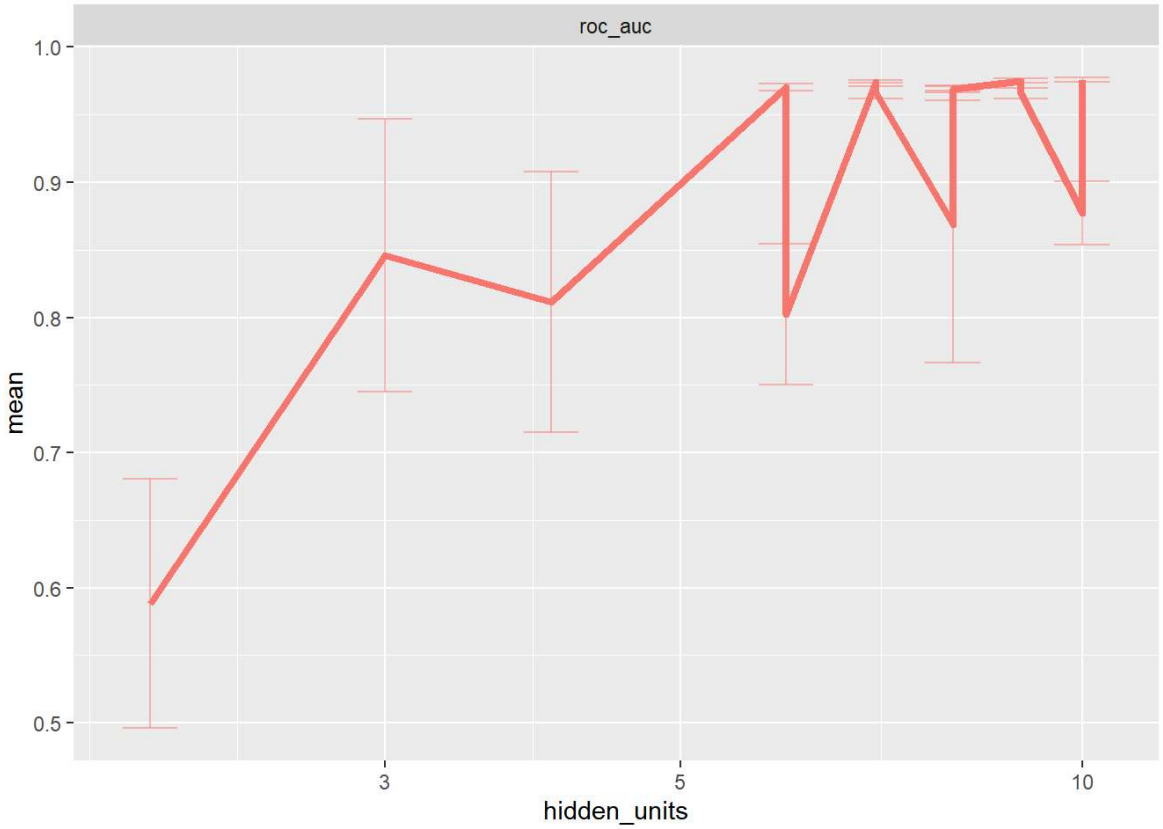
1-10 of 14 rows Previous 1 2 Next

```
nn_search_res %>%
  select_best("roc_auc")
```

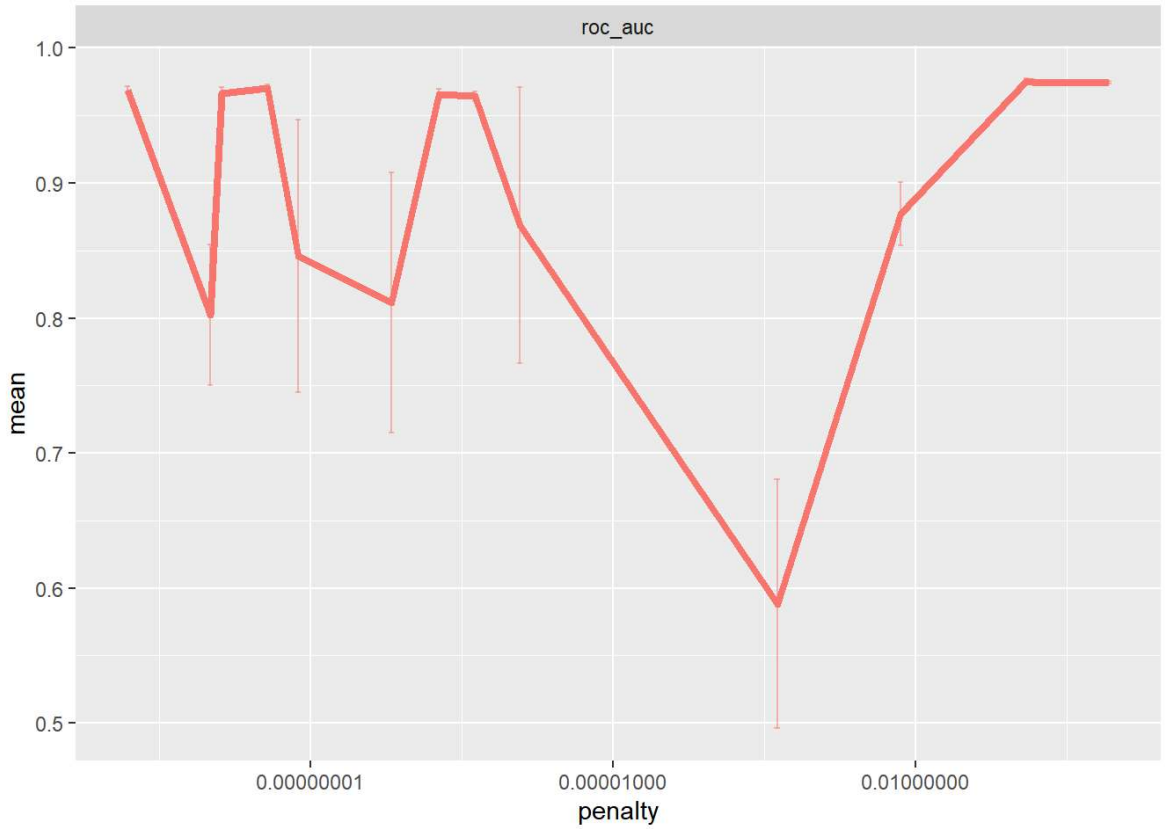
hidden_units	penalty	epochs	.config
<int>	<dbl>	<int>	<chr>
10	0.1251199	658	Iter4

1 row

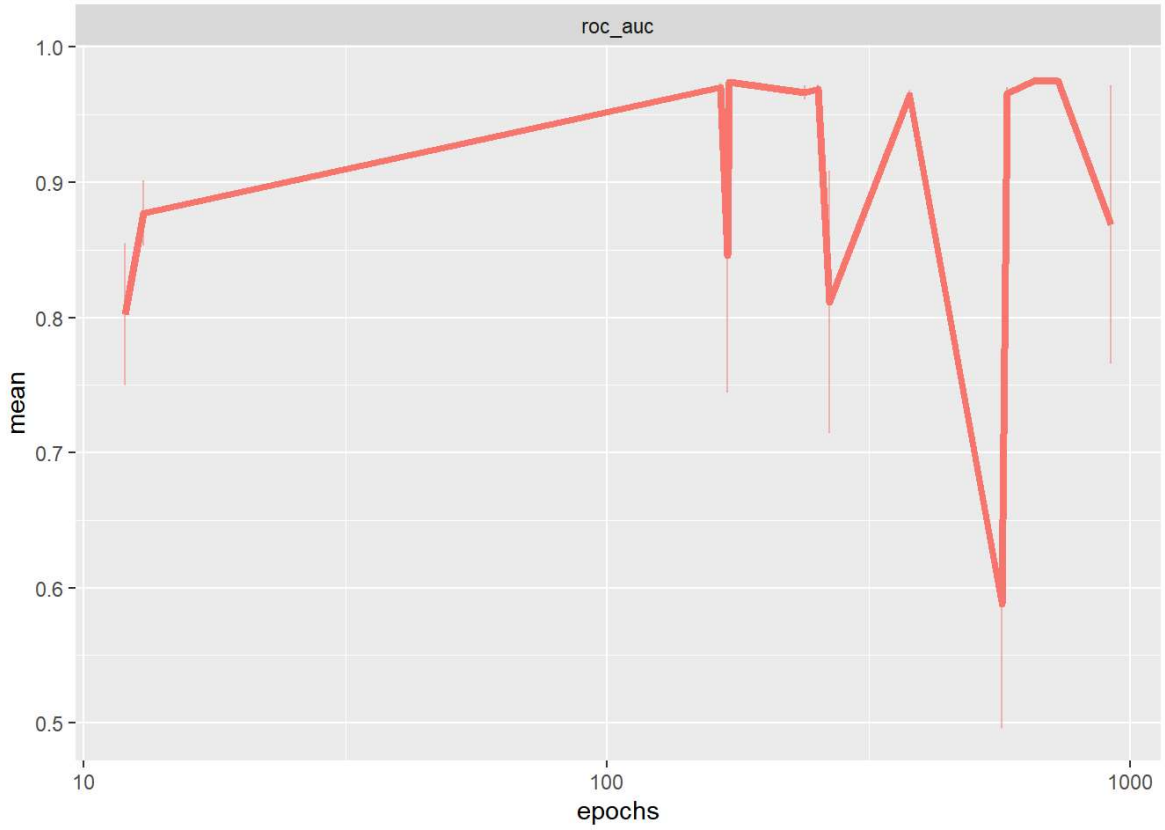
```
tune_graph <- function(parm){
# Graph of learning rate
nn_search_res %>%
  collect_metrics() %>%
  ggplot(aes(!!as.name(parm), mean, color = .metric)) +
  geom_errorbar(aes(
    ymin = mean - std_err,
    ymax = mean + std_err
  ),
  alpha = 0.5
  ) +
  geom_line(size = 1.5) +
  facet_wrap(~.metric, scales = "free", nrow = 2) +
  scale_x_log10() +
  theme(legend.position = "none")
}

tune_graph("hidden_units")
```



```
tune_graph("penalty")
```




```
tune_graph("epochs")
```



Final Fit

```
best_auc <- nn_search_res %>%
  select_best("roc_auc")

best_auc
```

hidden_units	penalty	epochs	.config
<int>	<dbl>	<int>	<chr>
10	0.1251199	658	Iter4

1 row

```
nn_wflow <- finalize_workflow(
  nn_wflow, best_auc
) %>%
  fit(train)
```

Score Neural Network Model

```
bind_cols(
  predict(nn_wflow, train, type="prob"),
  predict(nn_wflow, train, type="class"),
  train) %>%
  mutate(part = "train") -> scored_nn_train

bind_cols(
  predict(nn_wflow,test, type="prob"),
   predict(nn_wflow,test, type="class"),
  test) %>%
  mutate(part = "test") -> scored_nn_test
```

Neural Network Evaluation

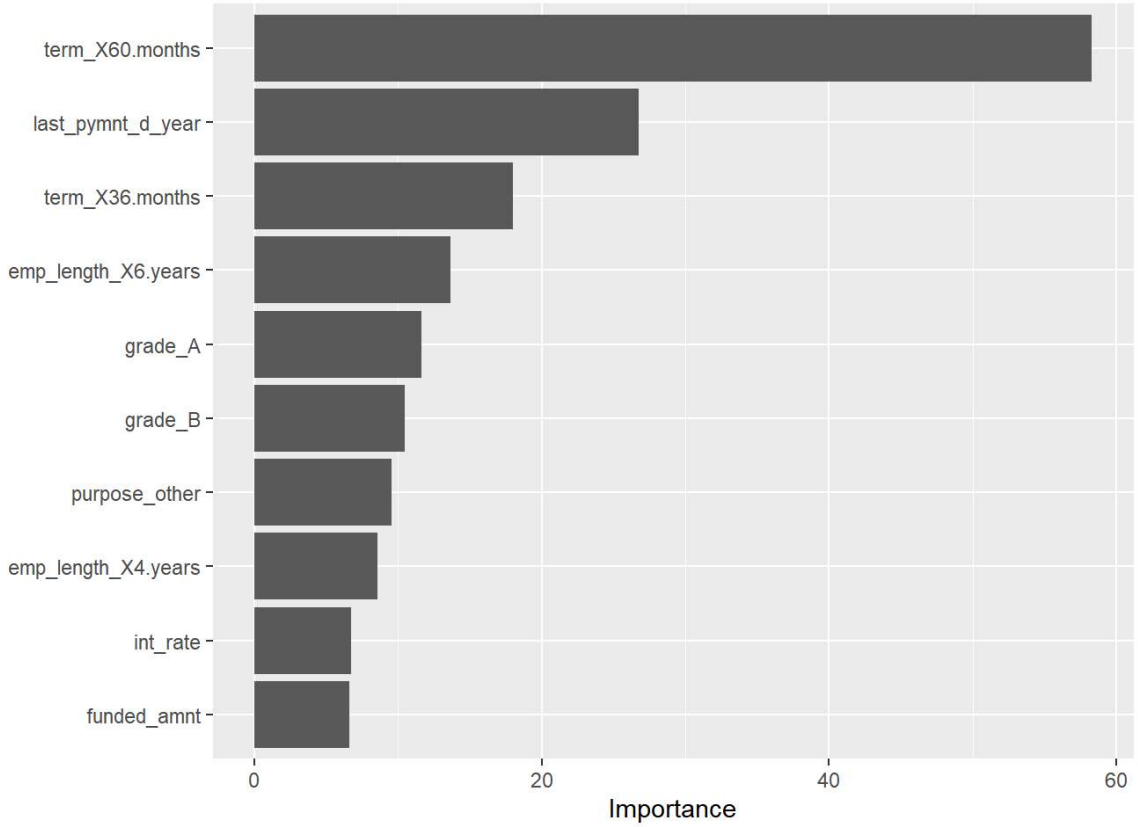
```
options(yardstick.event_first = FALSE)

# Variable Importance
nn_wflow %>%
  extract_fit_parsnip() %>%
  vi()
```

Variable	Importance
<chr>	<dbl>
term_X60.months	58.30648263
last_pymnt_d_year	26.77691513
term_X36.months	17.97824746

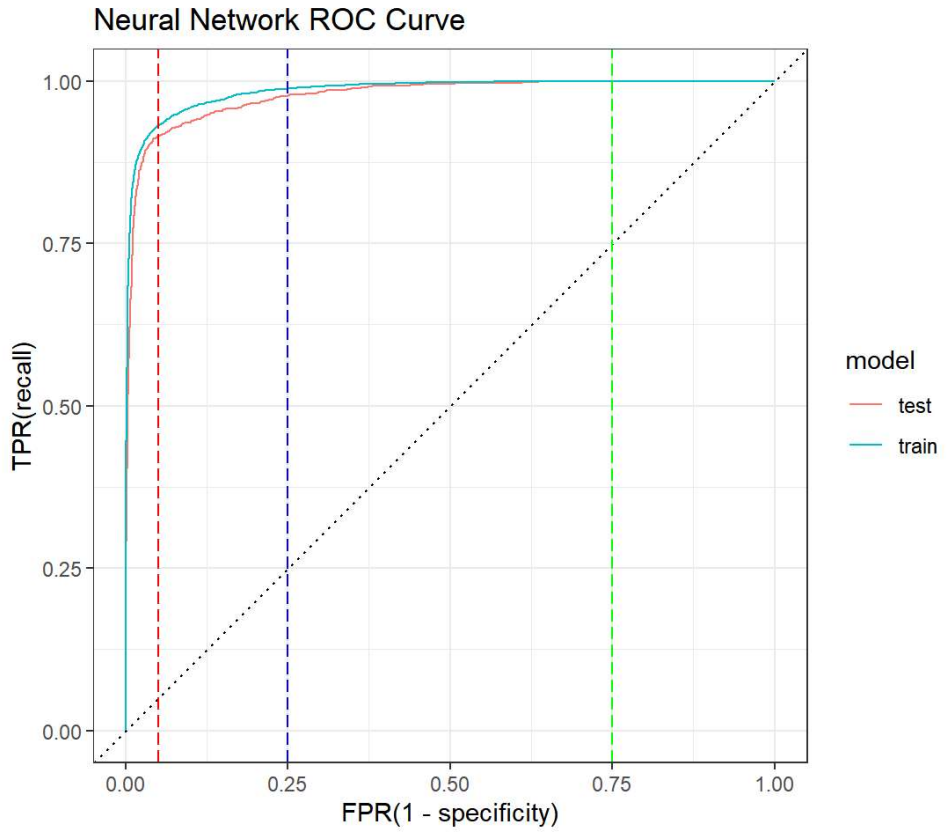
Variable	Importance
<chr>	<dbl>
emp_length_X6.years	13.66197398
grade_A	11.59995350
grade_B	10.48688356
purpose_other	9.52714520
emp_length_X4.years	8.55484792
int_rate	6.71747054
funded_amnt	6.61775193

1-10 of 30 rows Previous **1** 2 3 Next

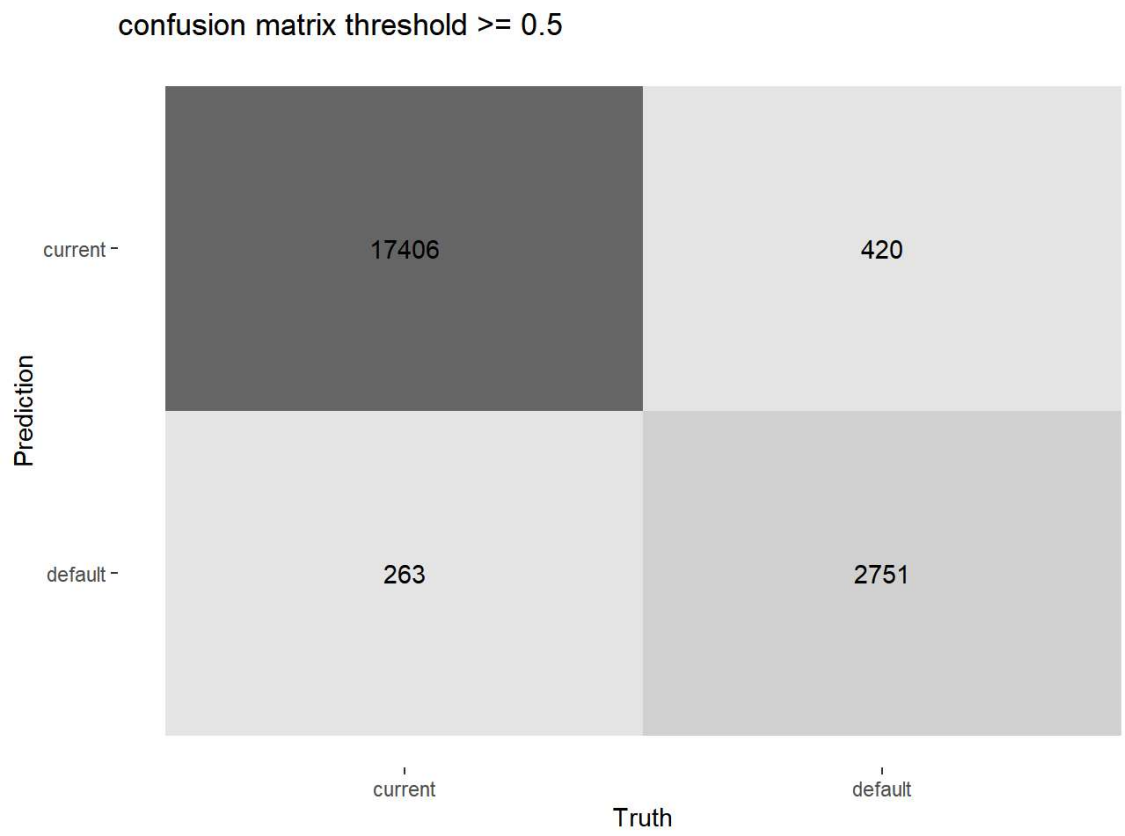
```
nn_wflow %>%
  extract_fit_parsnip() %>%
  vip()
```



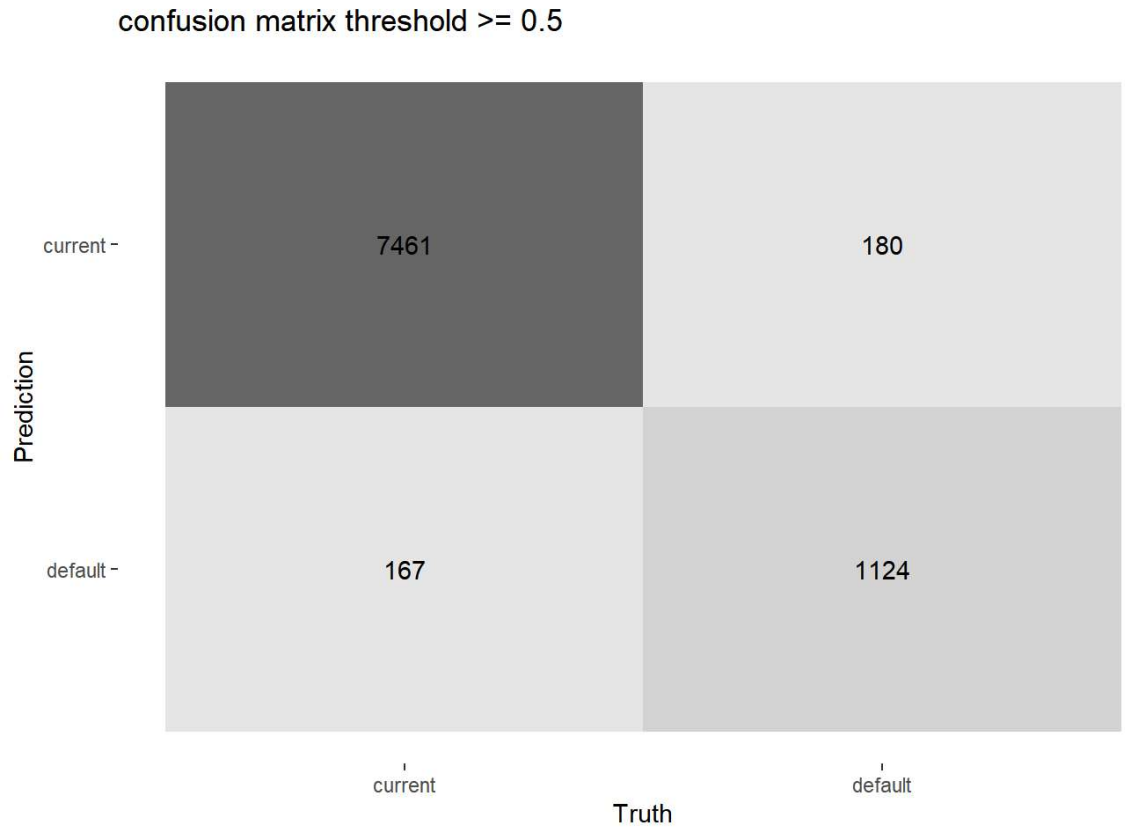
```
scored_nn_train <- scored_nn_train %>% mutate(part = "training")

scored_nn_test <- scored_nn_test %>% mutate(part = "testing")

scored_nn_train %>% mutate(model = "train") %>%
  bind_rows(scored_nn_test %>% mutate(model="test")) %>%
  group_by(model) %>%
  roc_curve(loan_status, .pred_default) %>%
  autoplot() +
  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 = "Neural Network ROC Curve" , x = "FPR(1 - specificity)", y = "TPR(recall)")
```




```
scored_nn_train %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current'))) %>%
   conf_mat(loan_status, estimate = predict_class) %>%
   autoplot(type = "heatmap") +
   labs(title="confusion matrix threshold >= 0.5")
```




```
scored_nn_test %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current'))) %>%
   conf_mat(loan_status, estimate = predict_class) %>%
   autoplot(type = "heatmap") +
   labs(title="confusion matrix threshold >= 0.5")
```



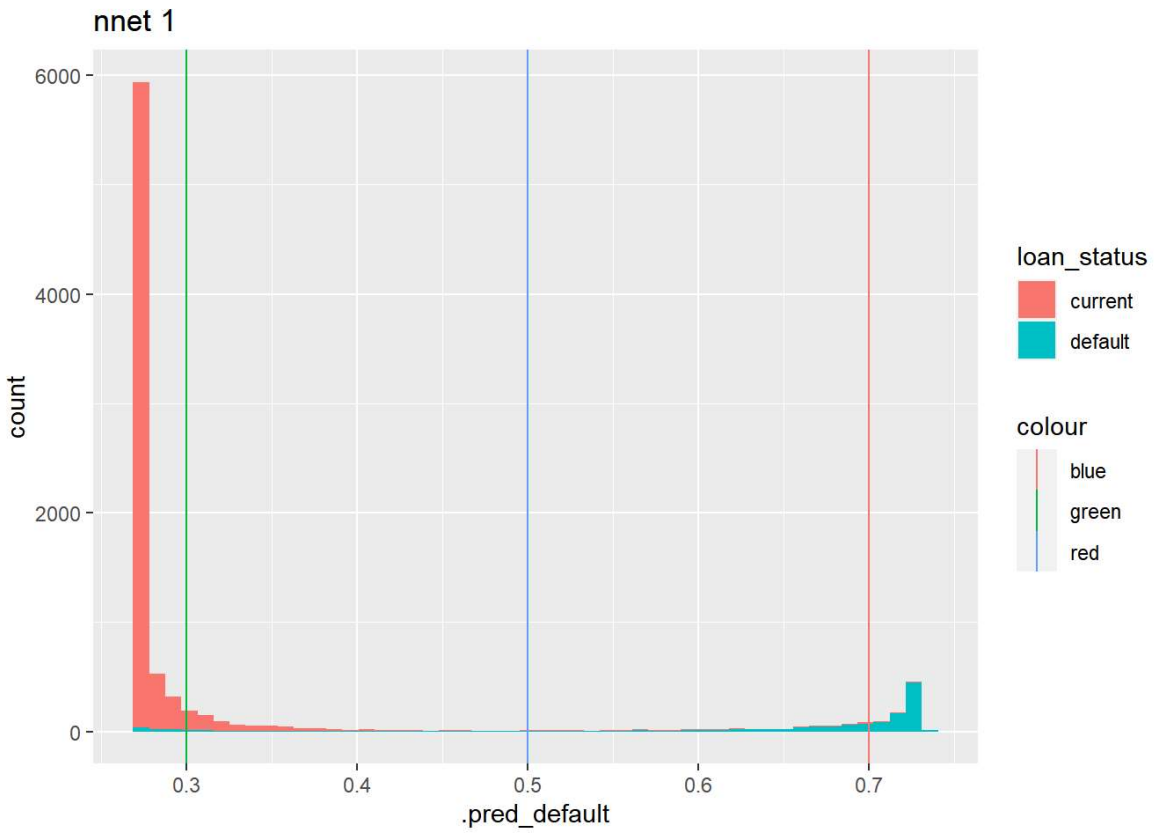
```
scored_nn_train_mutate <- scored_nn_train %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current')))

scored_nn_test_mutate <- scored_nn_test %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current')))

bind_rows(scored_nn_train, scored_nn_test) %>%
    group_by(part) %>%
    dplyr::select(loan_status, .pred_class) -> train_test

precision <- train_test %>%
  yardstick::precision(loan_status, .pred_class)

recall <- train_test %>%
  yardstick::recall(loan_status, .pred_class)

scored_nn_train %>%
    metrics(loan_status, .pred_default, estimate = .pred_class) %>%
    bind_rows(scored_nn_train_mutate %>% yardstick::precision(loan_status, predict_class)) %>%
    bind_rows(scored_nn_train_mutate %>% yardstick::recall(loan_status, predict_class)) %>%
    filter(.metric %in% c("accuracy", "roc_auc", "mn_log_loss", "precision", "recall")) %>%
    mutate(part="training") %>%
bind_rows(scored_nn_test %>%
    metrics(loan_status, .pred_default, estimate = .pred_class) %>%
    bind_rows(scored_nn_test_mutate %>% yardstick::precision(loan_status, predict_class)) %>%
    bind_rows(scored_nn_test_mutate %>% yardstick::recall(loan_status, predict_class)) %>%
    filter(.metric %in% c("accuracy", "roc_auc", "mn_log_loss", "precision", "recall")) %>%
    mutate(part="testing"))  %>%
bind_rows(
  bind_cols(precision, recall) %>%
    mutate(f1_score = 2*.estimate...4*.estimate...8/(.estimate...4 + .estimate...8)) %>%
    mutate(.metric = "F1_Score") %>%
    mutate(.estimator = "binary") %>%
    dplyr::select(part...1, .metric, .estimator, f1_score) %>%
    rename(c(".estimate" = "f1_score"))
) %>%
    arrange(desc(part))
```

.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
accuracy	binary	0.9672265	training
mn_log_loss	binary	0.3556071	training
roc_auc	binary	0.9848973	training
precision	binary	0.9127405	training
recall	binary	0.8675497	training
F1_Score	binary	0.8895715	training
accuracy	binary	0.9611509	testing

.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
mn_log_loss	binary	0.3607428	testing
roc_auc	binary	0.9770547	testing
precision	binary	0.8706429	testing

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

```
# Score Distribution
scored_nn_test %>%
  ggplot(aes(.pred_default, fill=loan_status)) +
  geom_histogram(bins=50) +
  geom_vline(aes(xintercept=.5, color="red")) +
  geom_vline(aes(xintercept=.3, color="green")) +
  geom_vline(aes(xintercept=.7, color="blue")) +
  labs(title = "nnet 1") -> scored_dist

print(scored_dist)
```



```
# operating range 0 - 10%
operating_range <- scored_nn_test %>%
  roc_curve(loan_status, .pred_default)  %>%
  mutate(
    fpr = round((1 - specificity), 3),
    tpr = round(sensitivity, 3),
    score_threshold =  round(.threshold, 5)
  ) %>%
  group_by(fpr) %>%
  summarise(threshold = round(mean(score_threshold),4),
            tpr = mean(tpr)) %>%
  filter(fpr <= 0.5)

  print(operating_range)
```

```
## # A tibble: 501 × 3
##      fpr threshold    tpr
##    <dbl>     <dbl>  <dbl>
##  1 0       Inf     0.0478
##  2 0.001     0.727 0.219
##  3 0.002     0.720 0.379
##  4 0.003     0.713 0.460
##  5 0.004     0.704 0.530
##  6 0.005     0.699 0.569
##  7 0.006     0.692 0.602
##  8 0.007     0.684 0.638
##  9 0.008     0.677 0.665
## 10 0.009     0.670 0.691
## # … with 491 more rows
```

```
# Precision Recall Chart
scored_nn_test %>%
  pr_curve(loan_status, .pred_default) %>%
  mutate(
    recall = round(recall, 2),
    .threshold = round(.threshold, 3),
    precision = round(precision, 3)
  ) %>%
  group_by(recall) %>%
  summarise(precision = max(precision),
            .threshold = min(.threshold))
```

recall	precision	.threshold
<dbl>	<dbl>	<dbl>
0.00	1.000	0.731
0.01	1.000	0.731
0.02	1.000	0.731
0.03	1.000	0.731

recall	precision	.threshold
<dbl>	<dbl>	<dbl>
0.04	0.983	0.731
0.05	0.986	0.730
0.06	0.987	0.730
0.07	0.980	0.730
0.08	0.982	0.730
0.09	0.983	0.730

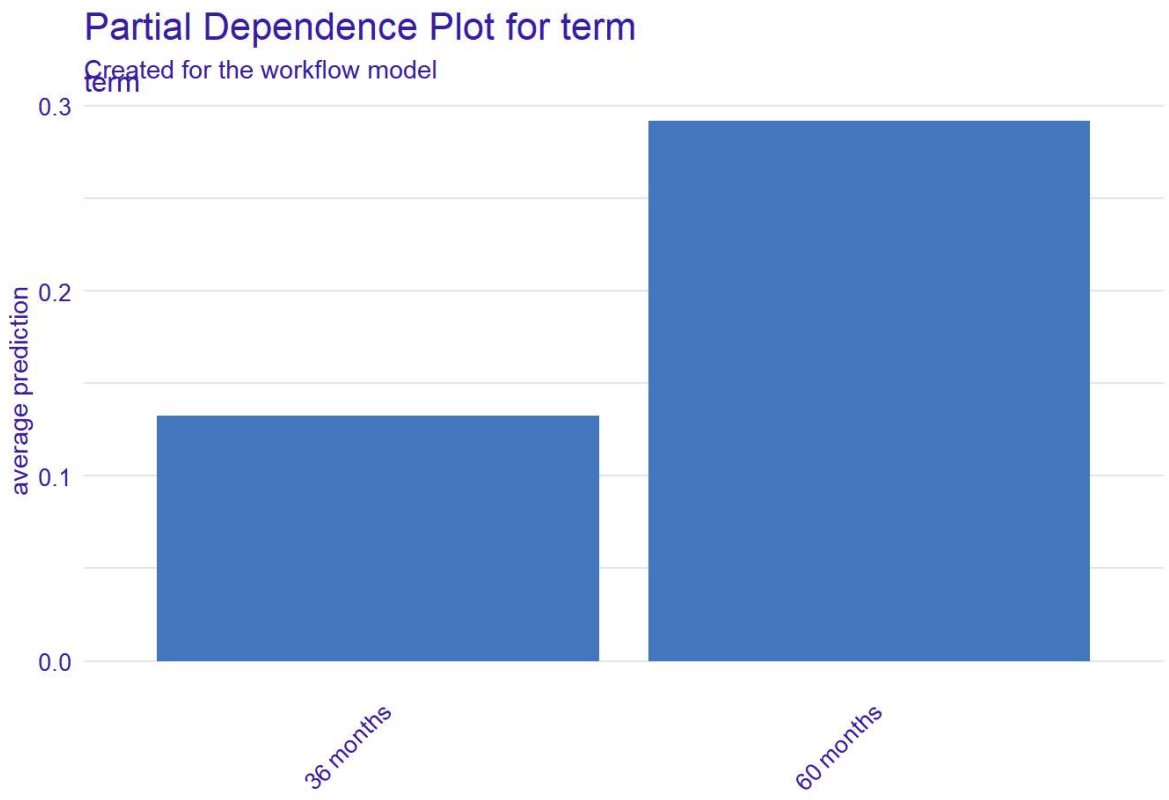
1-10 of 101 rows Previous 1 2 3 4 5 6 ... 11 Next

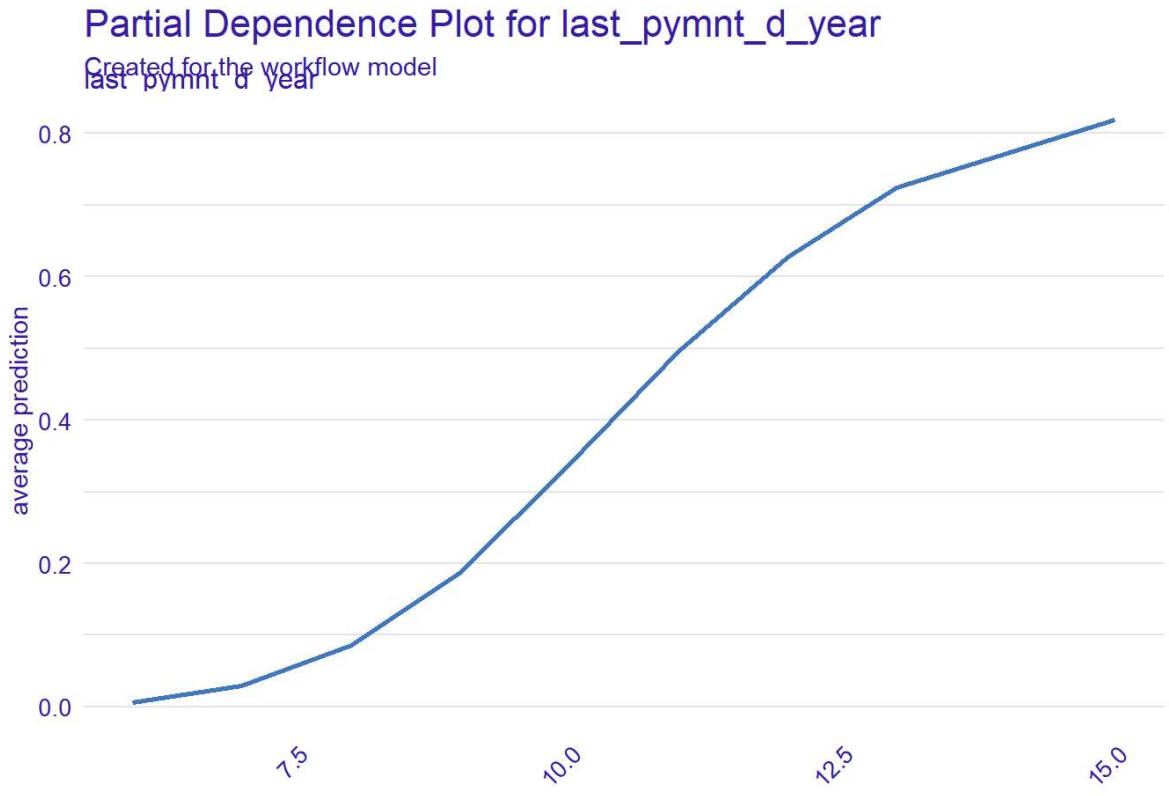
Partial Dependance Plot (Neural Network)

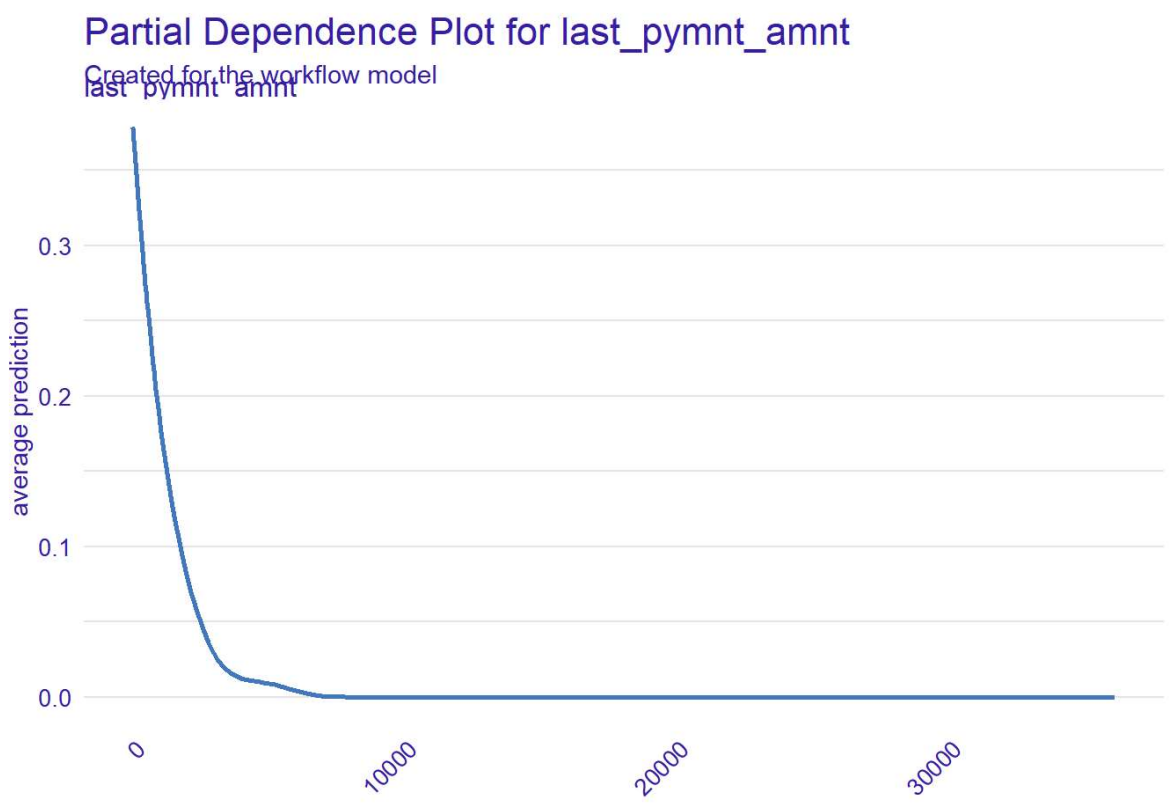
```
# create an explainer of a model


nn_explainer <- explain_tidymodels(
  logistic_wf,
  data = train ,
  y = train$loan_default ,
  verbose = TRUE
)
```

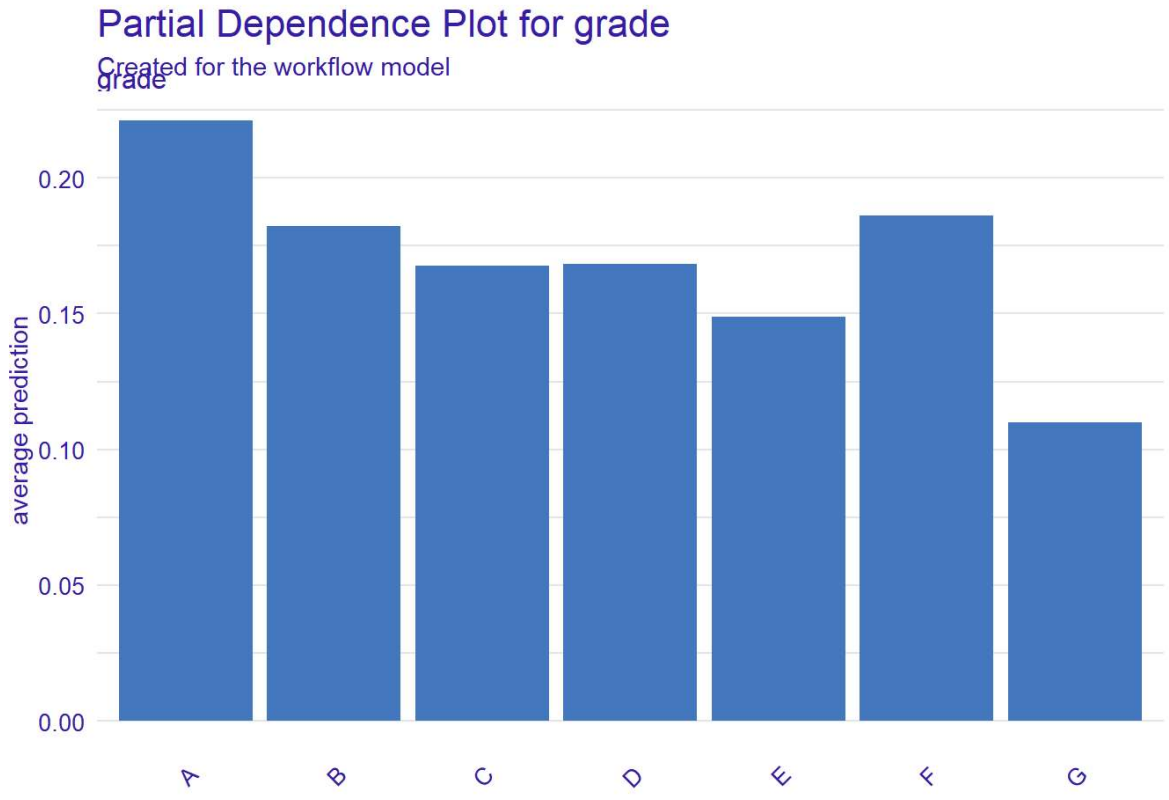
```
## Preparation of a new explainer is initiated
##   -> model label       :  workflow  (  default  )
##   -> data              :  20840  rows  32  cols
##   -> data              :  tibble converted into a data.frame
##   -> target variable   :  not specified! (  WARNING  )
##   -> predict function  :  yhat.workflow  will be used (  default  )
##   -> predicted values  :  No value for predict function target column. (  default  )
##   -> model_info        :  package tidymodels , ver. 1.0.0 , task classification (  default  )
##   -> model_info        :  Model info detected classification task but 'y' is a NULL .  (  WARNING  )
##   -> model_info        :  By deafult classification tasks supports only numercical 'y' parameter.
##   -> model_info        :  Consider changing to numerical vector with 0 and 1 values.
##   -> model_info        :  Otherwise I will not be able to calculate residuals or loss function.
##   -> predicted values  :  numerical, min =  0.00000000000000002220446 , mean =  0.1518757 , max =  0.9999777
##   -> residual function :  difference between y and yhat (  default  )
##   A new explainer has been created!
```

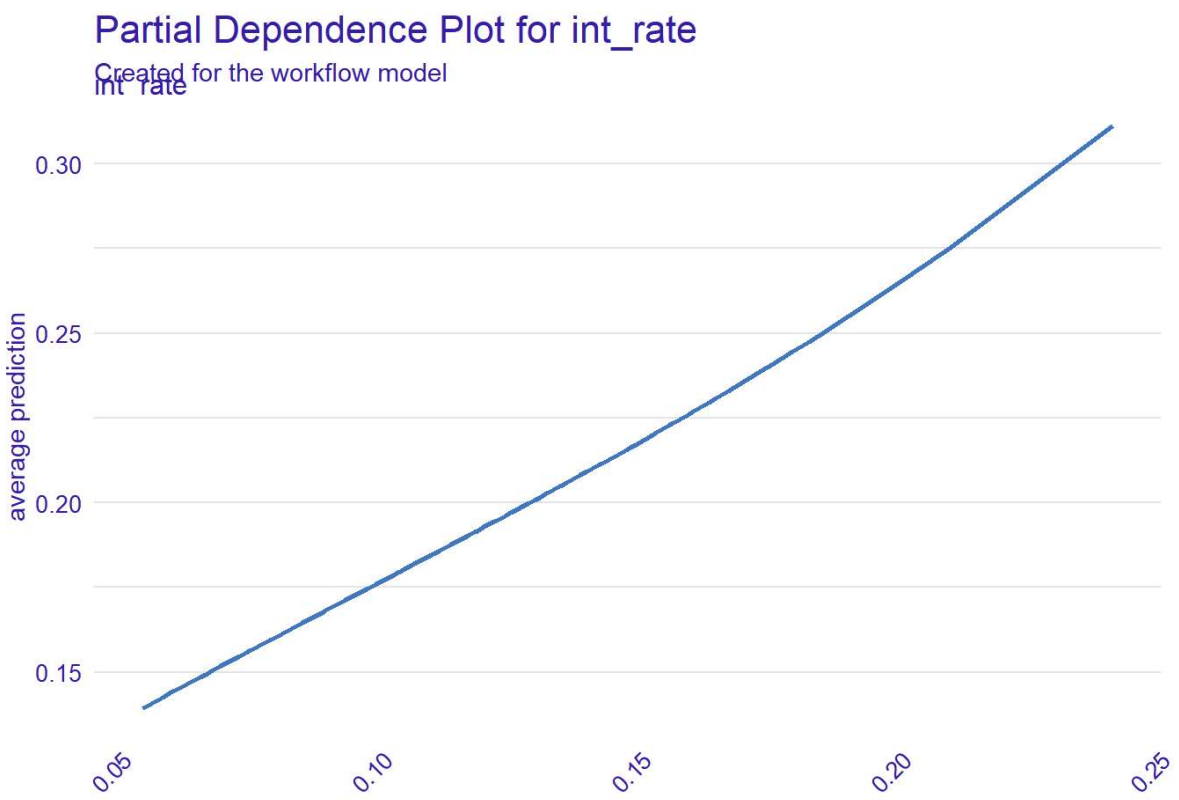
```
nn_variable <- c('term', 'last_pymnt_d_year', 'last_pymnt_amnt', 'grade', 'int_rate')

pdp <- function(m){

# create a profile of a single variable for a model

pdp_variable <- model_profile(
  nn_explainer,
  variables = as.character(m)
)

# Plot it

plot(pdp_variable) +
  ggtitle(paste0("Partial Dependence Plot for ", as.character(m))) +
  theme(axis.text.x=element_text(angle=45, hjust=1))

}

for (c in nn_variable){
    print(pdp(c))
}
```


Partial Dependence Plot for term
Created for the workflow model
term
average prediction
0.0
0.1
0.2
0.3
36 months
60 months


Partial Dependence Plot for last_pymnt_d_year
Created for the workflow model
last_pymnt_d_year
average prediction
0.0
0.2
0.4
0.6
0.8
7.5
10.0
12.5
15.0

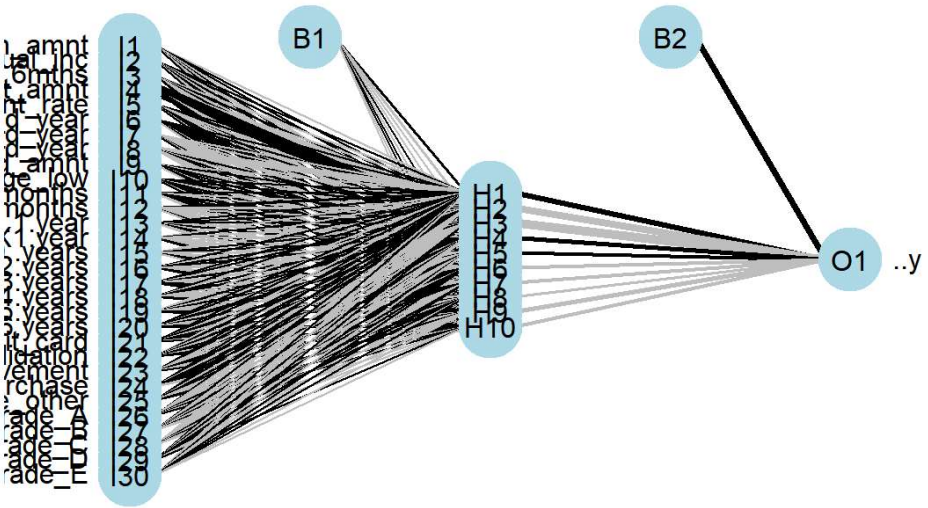

Partial Dependence Plot for last_pymnt_amnt
Created for the workflow model
last_pymnt_amnt
average prediction
0.3
0.2
0.1
0.0
0
10000
20000
30000


Partial Dependence Plot for grade
Created for the workflow model
grade
average prediction
0.20
0.15
0.10
0.05
0.00
A
B
C
D
E
F
G



Visualize Neural Networks

```
mod <- nn_wflow$fit$fit$fit
plotnet(mod)
```



Random Forest

Define Random Forest Model

```
kfold_splits <- vfold_cv(train, v=5)

rf_model <- rand_forest(trees=tune()) %>%
  set_engine("ranger", num.threads = 5, max.depth = 10, importance="permutation") %>%
  set_mode("classification")
```

Random Forest Workflow

```
rf_wflow <- workflow() %>%
  add_recipe(recipe) %>%
  add_model(rf_model)

rf_search_res <- rf_wflow %>%
  tune_bayes(
    resamples = kfold_splits,
    # Generate five at semi-random to start
    initial = 5,
    iter = 50,
    # How to measure performance?
    metrics = metric_set(yardstick::roc_auc),
    control = control_bayes(no_improve = 5, verbose = TRUE)
  )
```

```
##
```

```
## ❯  Generating a set of 5 initial parameter results
```

```
## ✓ Initialization complete
```

```
##
```

```
##
```

```
## ──── Iteration 1 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9696 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 1995 candidates
```

```
## i Predicted candidates
```

```
## i trees=1723
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.9695 (+/-0.00182)
```

```
##
```

```
## ── Iteration 2 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9696 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 1994 candidates
```

```
## i Predicted candidates
```

```
## i trees=1666
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.9695 (+/-0.00176)
```

```
##
```

```
## ── Iteration 3 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9696 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 1993 candidates
```

```
## i Predicted candidates
```

```
## i trees=1694
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.9693 (+/-0.00182)
```

```
##
```

```
## ── Iteration 4 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9696 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 1992 candidates
```

```
## i Predicted candidates
```

```
## i trees=711
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.9694 (+/-0.00175)
```

```
##
```

```
## ── Iteration 5 ──────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9696 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 1991 candidates
```

```
## i Predicted candidates
```

```
## i trees=460
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.9695 (+/-0.00162)
```

```
## ! No improvement for 5 iterations; returning current results.
```

Final Fit Random Forest

```
highest_rf_auc <- rf_search_res %>%
  select_best("roc_auc")

highest_rf_auc
```

trees	.config
<int>	<chr>
1682	Preprocessor1_Model2

1 row

```
rf_wflow <- finalize_workflow(
  rf_wflow, highest_rf_auc
) %>%
  fit(train)
```

Score Random Forest Model

```
  # score training
  predict(rf_wflow, train, type="prob") %>%
    bind_cols(predict(rf_wflow, train, type="class")) %>%
    bind_cols(., train) -> scored_train_rf

  # score testing
  predict(rf_wflow, test, type="prob") %>%
      bind_cols(predict(rf_wflow, test, type="class")) %>%
      bind_cols(., test) -> scored_test_rf
```

Random Forest Evaluation

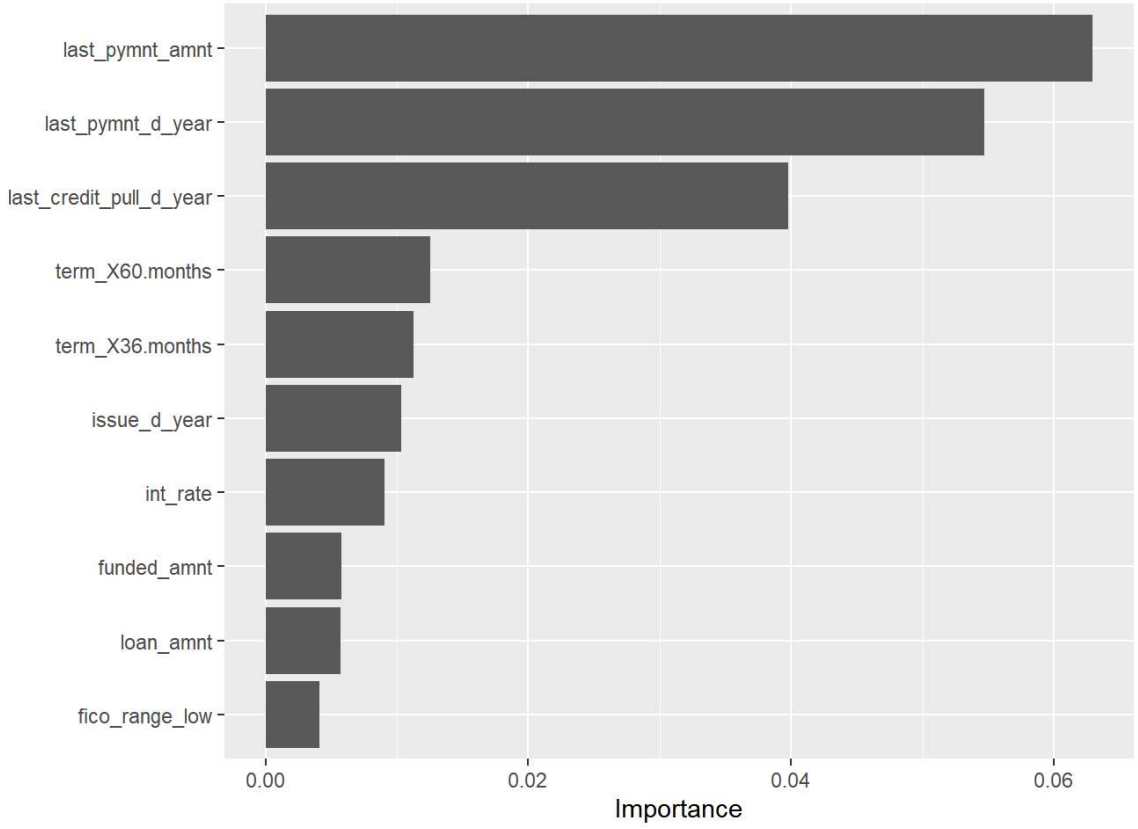
```
options(yardstick.event_first = FALSE)

# Variable Importance
rf_wflow %>%
  extract_fit_parsnip() %>%
  vi()
```

Variable	Importance
<chr>	<dbl>
last_pymnt_amnt	0.062932913579
last_pymnt_d_year	0.054697154403
last_credit_pull_d_year	0.039801350910
term_X60.months	0.012575861286

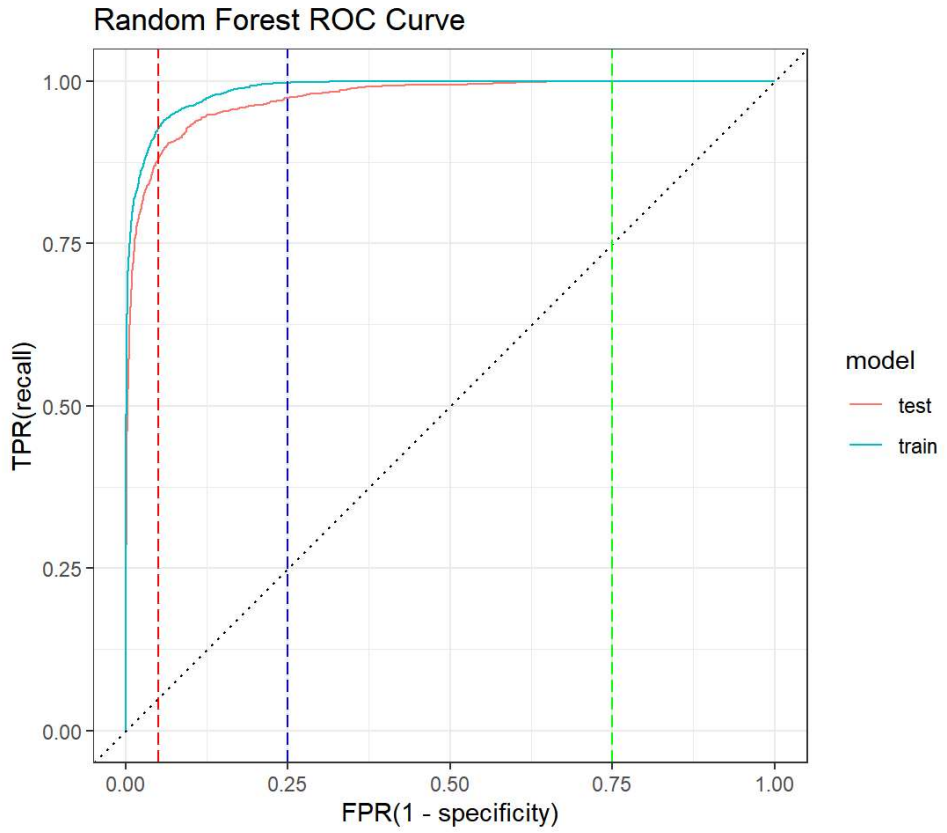
Variable	Importance
<chr>	<dbl>
term_X36.months	0.011241030418
issue_d_year	0.010301794910
int_rate	0.009071465987
funded_amnt	0.005753499349
loan_amnt	0.005745099891
fico_range_low	0.004130807859

1-10 of 30 rows Previous **1** 2 3 Next

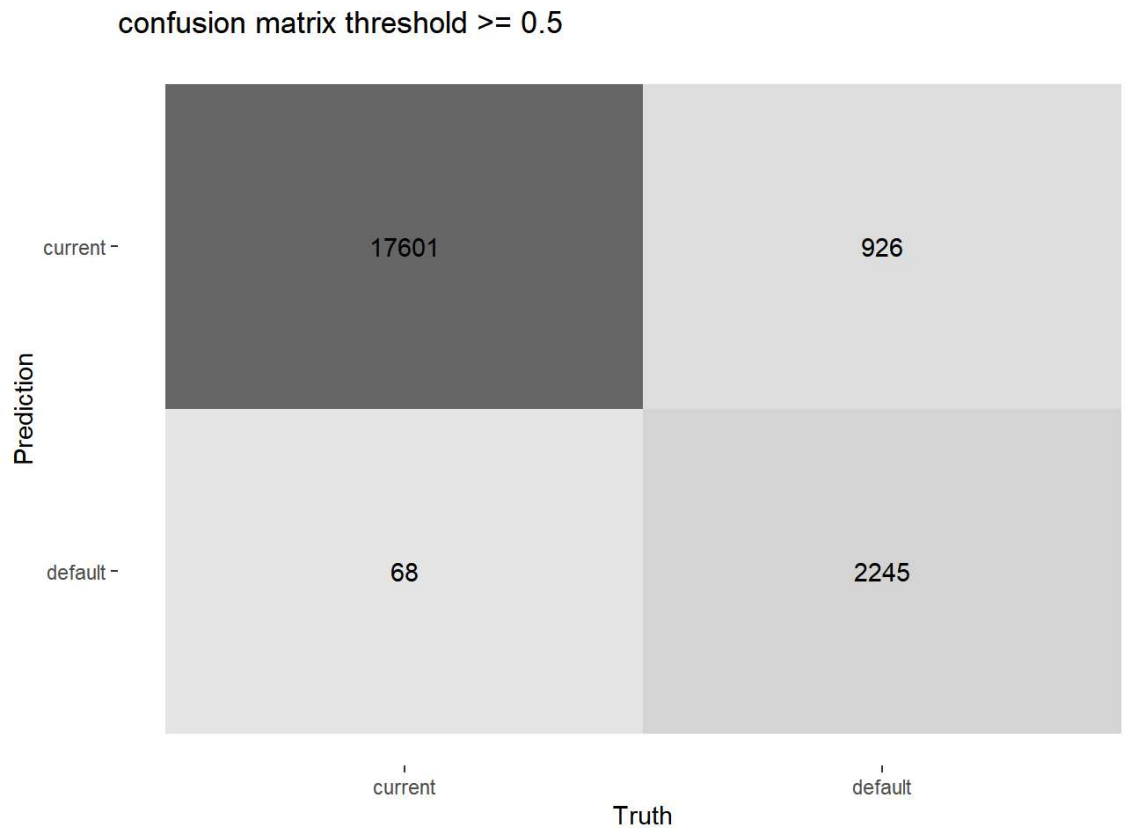
```
rf_wflow %>%
  extract_fit_parsnip() %>%
  vip()
```



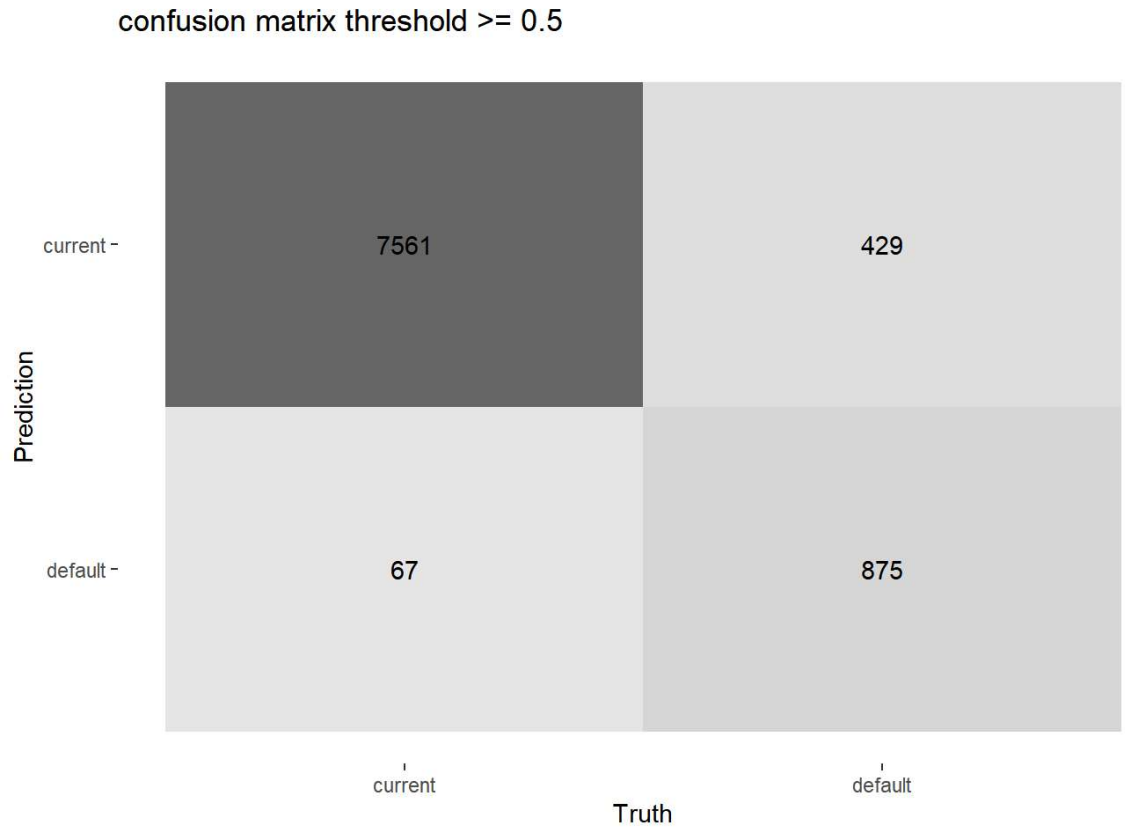
```
scored_train_rf <- scored_train_rf %>% mutate(part = "training")

scored_test_rf <- scored_test_rf %>% mutate(part = "testing")

scored_train_rf %>% mutate(model = "train") %>%
  bind_rows(scored_test_rf %>% mutate(model="test")) %>%
  group_by(model) %>%
  roc_curve(loan_status, .pred_default) %>%
  autoplot() +
  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 = "Random Forest ROC Curve" , x = "FPR(1 - specificity)", y = "TPR(recall)")
```




```
scored_train_rf %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current'))) %>%
   conf_mat(loan_status, estimate = predict_class) %>%
   autoplot(type = "heatmap") +
   labs(title="confusion matrix threshold >= 0.5")
```




```
scored_test_rf %>%
    mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current'))) %>%
    conf_mat(loan_status, estimate = predict_class) %>%
    autoplot(type = "heatmap") +
    labs(title="confusion matrix threshold >= 0.5")
```



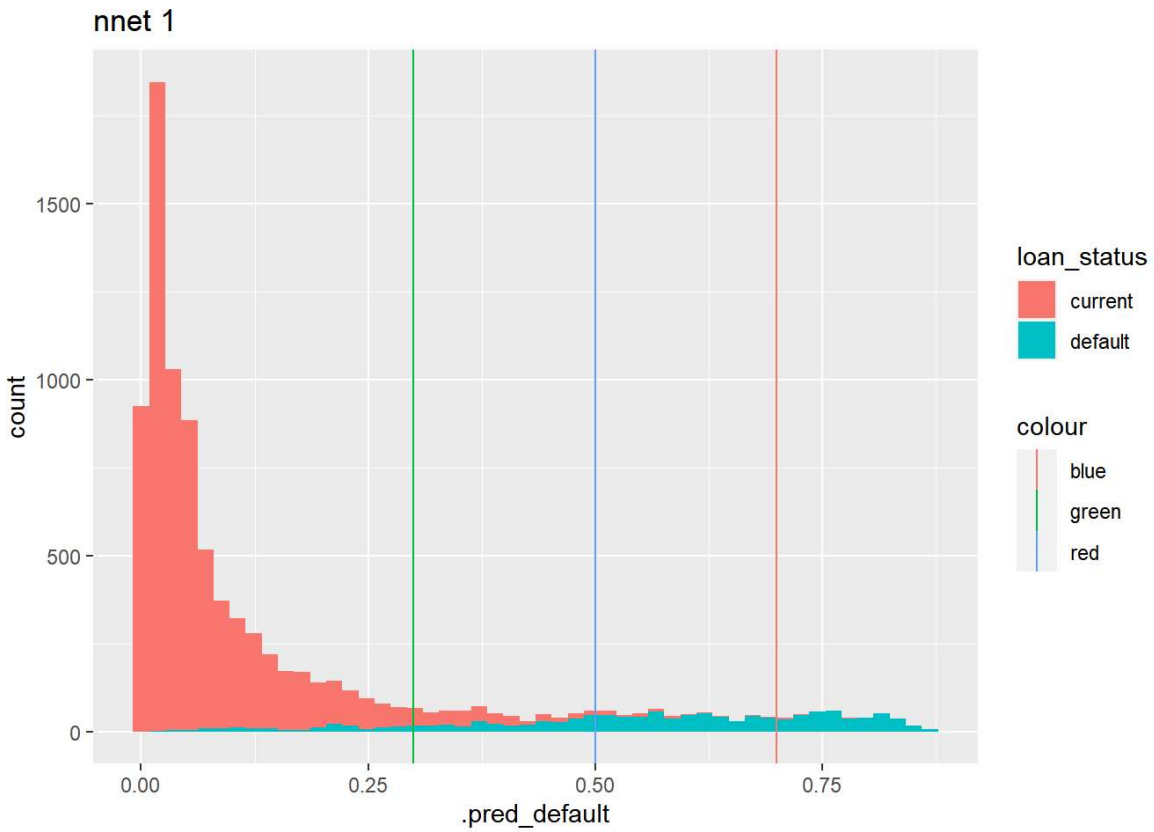
```
scored_train_rf_mutate <- scored_train_rf %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current')))

scored_test_rf_mutate <- scored_test_rf %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current')))

bind_rows(scored_train_rf, scored_test_rf) %>%
    group_by(part) %>%
    dplyr::select(loan_status, .pred_class) -> train_test

precision <- train_test %>%
  yardstick::precision(loan_status, .pred_class)

recall <- train_test %>%
  yardstick::recall(loan_status, .pred_class)

scored_train_rf %>%
    metrics(loan_status, .pred_default, estimate = .pred_class) %>%
    bind_rows(scored_train_rf_mutate %>% yardstick::precision(loan_status, predict_class)) %>%
    bind_rows(scored_train_rf_mutate %>% yardstick::recall(loan_status, predict_class)) %>%
    filter(.metric %in% c("accuracy", "roc_auc", "mn_log_loss", "precision", "recall")) %>%
    mutate(part="training") %>%
bind_rows(scored_test_rf %>%
    metrics(loan_status, .pred_default, estimate = .pred_class) %>%
    bind_rows(scored_train_rf_mutate %>% yardstick::precision(loan_status, predict_class)) %>%
    bind_rows(scored_train_rf_mutate %>% yardstick::recall(loan_status, predict_class)) %>%
    filter(.metric %in% c("accuracy", "roc_auc", "mn_log_loss", "precision", "recall")) %>%
    mutate(part="testing"))  %>%
bind_rows(
  bind_cols(precision, recall) %>%
    mutate(f1_score = 2*.estimate...4*.estimate...8/(.estimate...4 + .estimate...8)) %>%
    mutate(.metric = "F1_Score") %>%
    mutate(.estimator = "binary") %>%
    dplyr::select(part...1, .metric, .estimator, f1_score) %>%
    rename(c(".estimate" = "f1_score"))
) %>%
    arrange(desc(part))
```

.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
accuracy	binary	0.9523033	training
mn_log_loss	binary	0.1623706	training
roc_auc	binary	0.9867277	training
precision	binary	0.9706010	training
recall	binary	0.7079786	training
F1_Score	binary	0.8187454	training
accuracy	binary	0.9444693	testing

.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
mn_log_loss	binary	0.1797455	testing
roc_auc	binary	0.9732740	testing
precision	binary	0.9706010	testing

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

```
# Score Distribution
scored_test_rf %>%
  ggplot(aes(.pred_default, fill=loan_status)) +
  geom_histogram(bins=50) +
  geom_vline(aes(xintercept=.5, color="red")) +
  geom_vline(aes(xintercept=.3, color="green")) +
  geom_vline(aes(xintercept=.7, color="blue")) +
  labs(title = "nnet 1") -> scored_dist

print(scored_dist)
```



```
# operating range 0 - 10%
operating_range <- scored_test_rf %>%
  roc_curve(loan_status, .pred_default)  %>%
  mutate(
    fpr = round((1 - specificity), 3),
    tpr = round(sensitivity, 3),
    score_threshold =  round(.threshold, 5)
  ) %>%
  group_by(fpr) %>%
  summarise(threshold = round(mean(score_threshold),4),
            tpr = mean(tpr)) %>%
  filter(fpr <= 0.5)

  print(operating_range)
```

```
## # A tibble: 501 × 3
##      fpr threshold    tpr
##    <dbl>     <dbl>  <dbl>
##  1 0       Inf     0.0703
##  2 0.001     0.730 0.235
##  3 0.002     0.635 0.407
##  4 0.003     0.585 0.501
##  5 0.004     0.567 0.536
##  6 0.005     0.554 0.565
##  7 0.006     0.534 0.600
##  8 0.007     0.515 0.635
##  9 0.008     0.505 0.656
## 10 0.009     0.496 0.676
## # … with 491 more rows
```

```
# Precision Recall Chart
scored_test_rf %>%
  pr_curve(loan_status, .pred_default) %>%
  mutate(
    recall = round(recall, 2),
    .threshold = round(.threshold, 3),
    precision = round(precision, 3)
  ) %>%
  group_by(recall) %>%
  summarise(precision = max(precision),
            .threshold = min(.threshold))
```

recall	precision	.threshold
<dbl>	<dbl>	<dbl>
0.00	1.000	0.860
0.01	1.000	0.846
0.02	1.000	0.838
0.03	1.000	0.831

recall	precision	.threshold
<dbl>	<dbl>	<dbl>
0.04	1.000	0.825
0.05	1.000	0.822
0.06	1.000	0.818
0.07	1.000	0.811
0.08	0.991	0.807
0.09	0.992	0.803

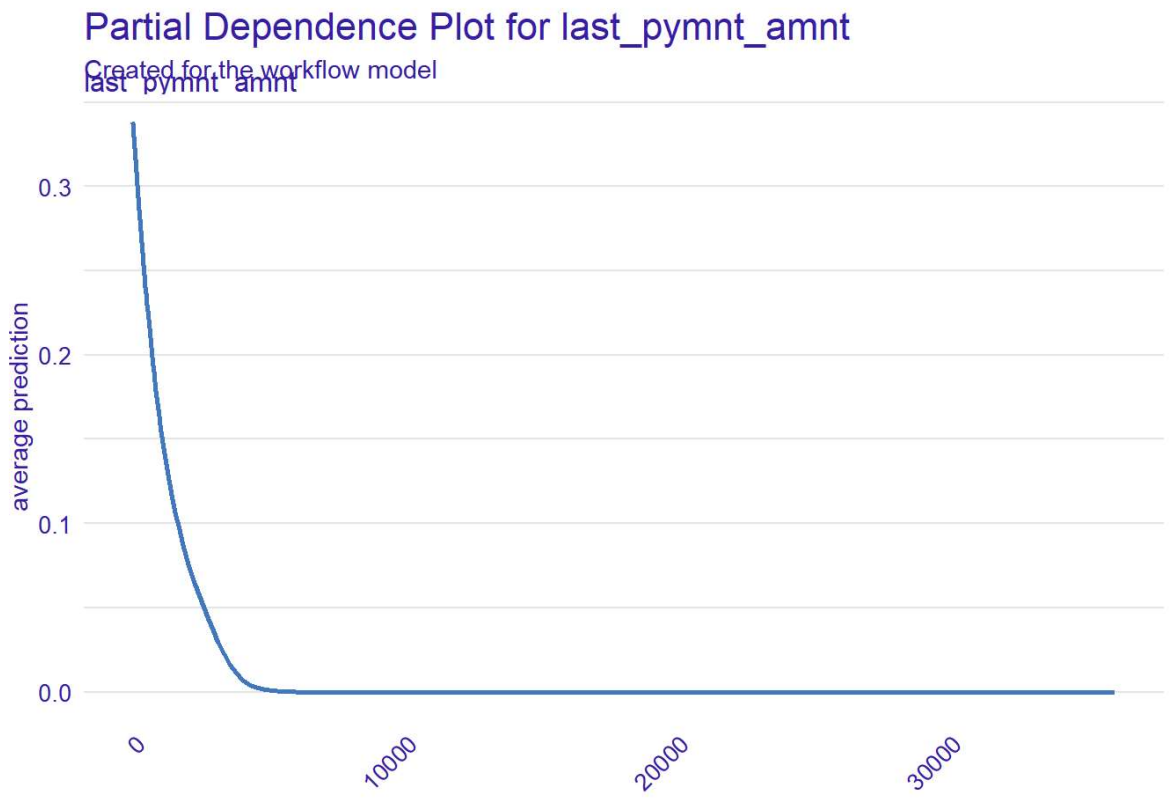
1-10 of 101 rows Previous **1** 2 3 4 5 6 ... 11 Next

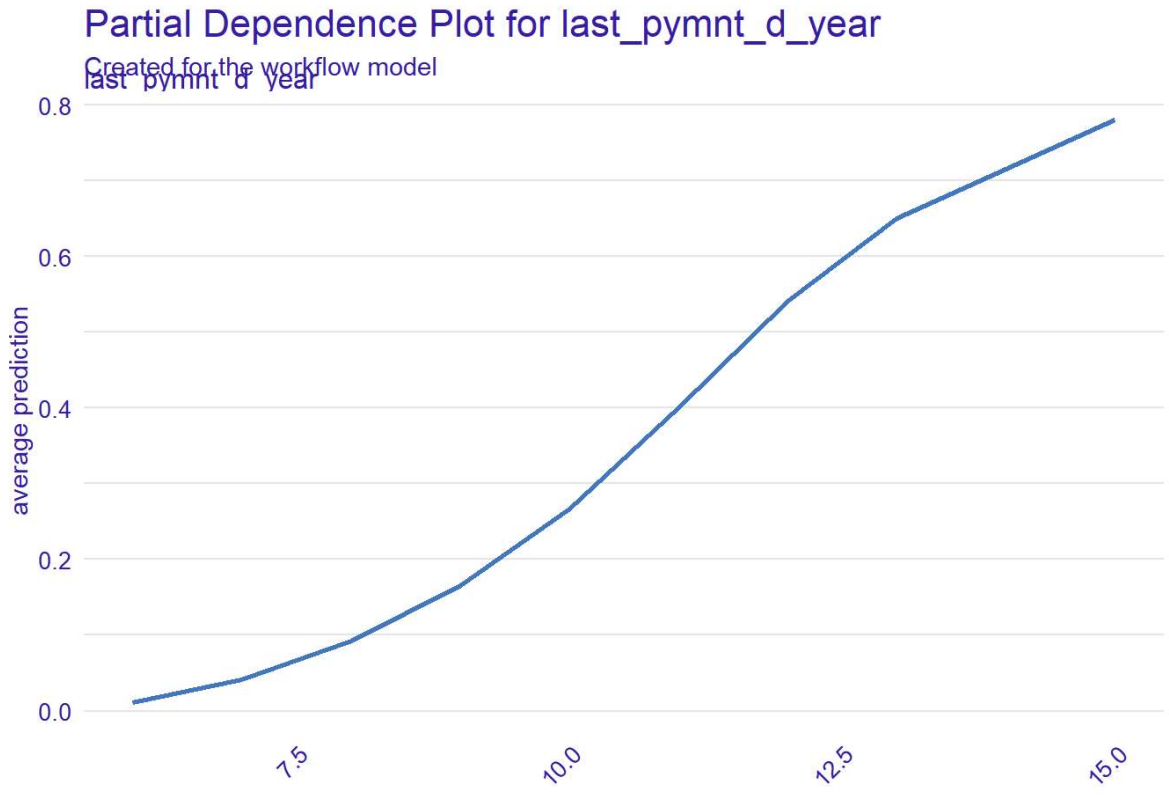
Partial Dependance Plot (Random Forest)

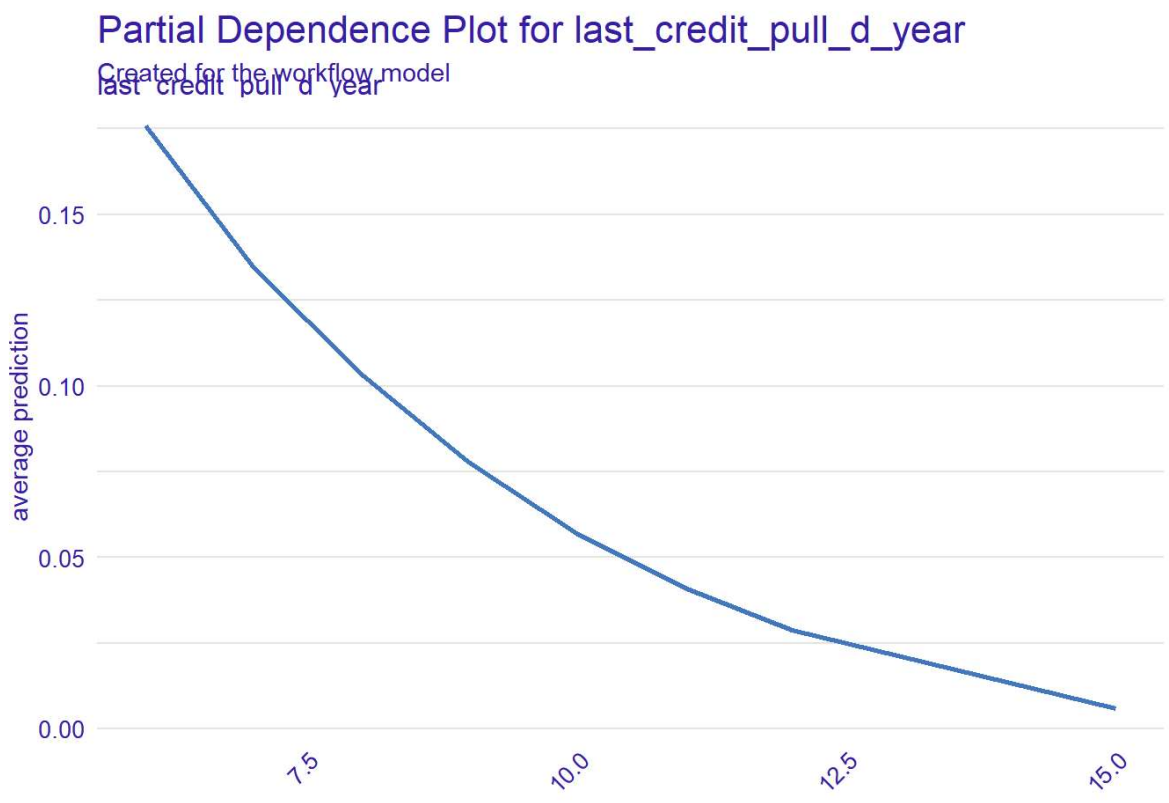
```
# create an explainer of a model


rf_explainer <- explain_tidymodels(
  logistic_wf,
  data = train ,
  y = train$loan_default ,
  verbose = TRUE
)
```

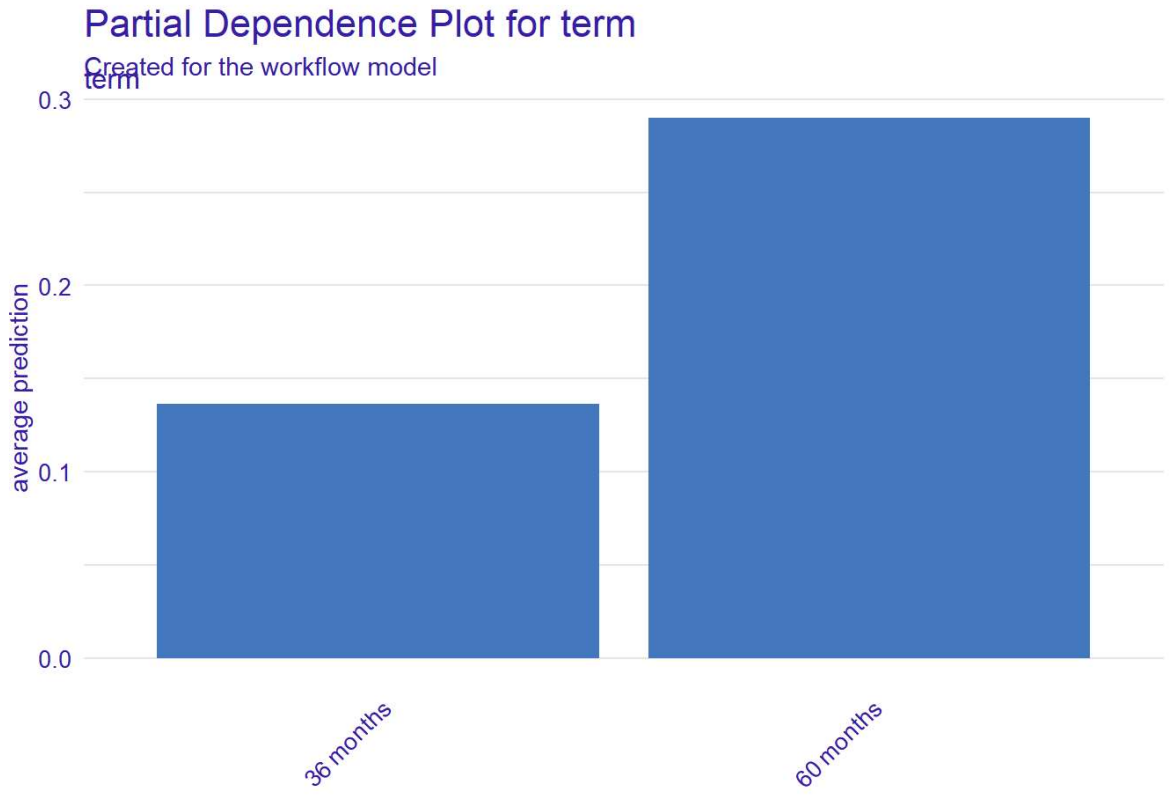
```
## Preparation of a new explainer is initiated
##   -> model label       :  workflow  (  default  )
##   -> data              :  20840  rows  32  cols
##   -> data              :  tibble converted into a data.frame
##   -> target variable   :  not specified! (  WARNING  )
##   -> predict function  :  yhat.workflow  will be used (  default  )
##   -> predicted values  :  No value for predict function target column. (  default  )
##   -> model_info        :  package tidymodels , ver. 1.0.0 , task classification (  default  )
##   -> model_info        :  Model info detected classification task but 'y' is a NULL .  (  WARNING  )
##   -> model_info        :  By deafult classification tasks supports only numercical 'y' parameter.
##   -> model_info        :  Consider changing to numerical vector with 0 and 1 values.
##   -> model_info        :  Otherwise I will not be able to calculate residuals or loss function.
##   -> predicted values  :  numerical, min =  0.00000000000000002220446 , mean =  0.1518757 , max =  0.9999777
##   -> residual function :  difference between y and yhat (  default  )
##   A new explainer has been created!
```

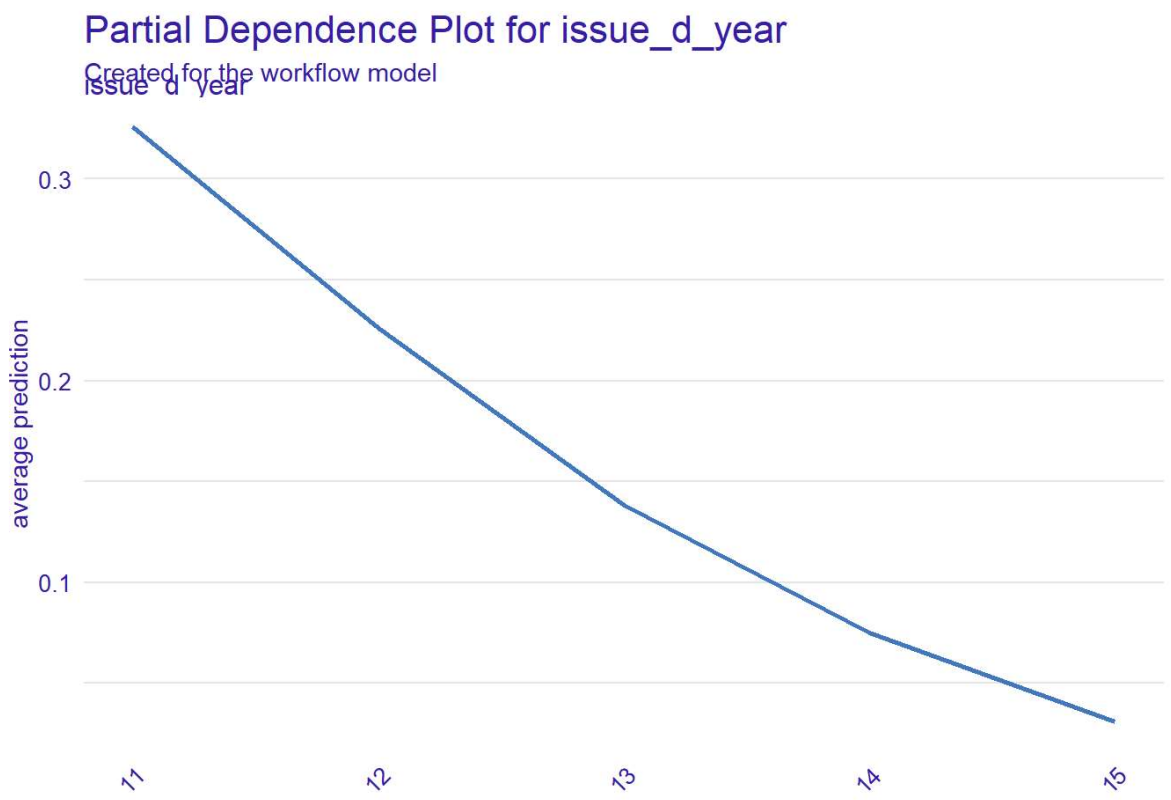
```
rf_variable <- c('last_pymnt_amnt', 'last_pymnt_d_year', 'last_credit_pull_d_year', 'term', 'issue_d_year')

pdp <- function(m){

# create a profile of a single variable for a model

pdp_variable <- model_profile(
  rf_explainer,
  variables = as.character(m)
)

# Plot it

plot(pdp_variable) +
  ggtitle(paste0("Partial Dependence Plot for ", as.character(m))) +
  theme(axis.text.x=element_text(angle=45, hjust=1))

}

for (c in rf_variable){
    print(pdp(c))
}
```


Partial Dependence Plot for last_pymnt_amnt
Created for the workflow model
last_pymnt_amnt
average prediction
0.3
0.2
0.1
0.0
0
10000
20000
30000


Partial Dependence Plot for last_pymnt_d_year
Created for the workflow model
last_pymnt_d_year
average prediction
0.8
0.6
0.4
0.2
0.0
7.5
10.0
12.5
15.0


Partial Dependence Plot for last_credit_pull_d_year
Created for the workflow model
last_credit_pull_d_year
average prediction
0.15
0.10
0.05
0.00
7.5
10.0
12.5
15.0


Partial Dependence Plot for term
Created for the workflow model
term
average prediction
0.3
0.2
0.1
0.0
36 months
60 months



XGBoost

XGBoost Model Buiding

Here we want to TUNE our XGB model using the Bayes method.

```
xgb_model <- boost_tree(trees = tune(),
                        learn_rate = tune(),
                        tree_depth = tune()) %>%
  set_engine("xgboost",
             importance="permutation") %>%
  set_mode("classification")

xgb_wflow <- workflow() %>%
  add_recipe(recipe) %>%
  add_model(xgb_model)

xgb_search_res <- xgb_wflow %>%
  tune_bayes(
    resamples = kfold_splits,
    # Generate five at semi-random to start
    initial = 5,
    iter = 50,
    metrics = metric_set(yardstick::roc_auc),
    control = control_bayes(no_improve = 5, verbose = TRUE)
  )
```

```
##
```

```
## ❯ Generating a set of 5 initial parameter results
```

```
## ✓ Initialization complete
```

```
##
```

```
##
```

```
## ── Iteration 1 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9798 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i trees=1668, tree_depth=8, learn_rate=0.306
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.978 (+/-0.000737)
```

```
##
```

```
## ── Iteration 2 ──────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9798 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i trees=1968, tree_depth=11, learn_rate=0.00122
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.9769 (+/-0.00115)
```

```
##
```

```
## ─── Iteration 3 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9798 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i trees=1426, tree_depth=4, learn_rate=0.00318
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.9741 (+/-0.00133)
```

```
##
```

```
## ──── Iteration 4 ──────────────────────────────────────────────────────────────────────────────────────────────────────
──────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9798 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i trees=1777, tree_depth=9, learn_rate=0.001
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.9759 (+/-0.00107)
```

```
##
```

```
## ── Iteration 5 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      roc_auc=0.9798 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i trees=1827, tree_depth=10, learn_rate=0.076
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    roc_auc=0.9789 (+/-0.00114)
```

```
## ! No improvement for 5 iterations; returning current results.
```

Final Fit XGBoost

```
highest_xgb_auc <- xgb_search_res %>%
  select_best("roc_auc")

highest_xgb_auc
```

trees	tree_depth	learn_rate	.config
<int>	<int>	<dbl>	<chr>
1518	10	0.03115321	Preprocessor1_Model4

1 row

```
xgb_wflow <- finalize_workflow(
  xgb_wflow, highest_xgb_auc
) %>%
  fit(train)
```

```
## [20:11:19] WARNING: amalgamation/../src/learner.cc:627:
## Parameters: { "importance" } might not be used.
##
##   This could be a false alarm, with some parameters getting used by language bindings but
##   then being mistakenly passed down to XGBoost core, or some parameter actually being used
##   but getting flagged wrongly here. Please open an issue if you find any such cases.
```

Score XGBoost Model

```
# -- score training
predict(xgb_wflow, train, type="prob") %>%
  bind_cols(predict(xgb_wflow, train, type="class")) %>%
  bind_cols(., train) -> scored_train_xgb

# -- score testing
predict(xgb_wflow, test, type="prob") %>%
    bind_cols(predict(xgb_wflow, test, type="class")) %>%
    bind_cols(., test) -> scored_test_xgb
```

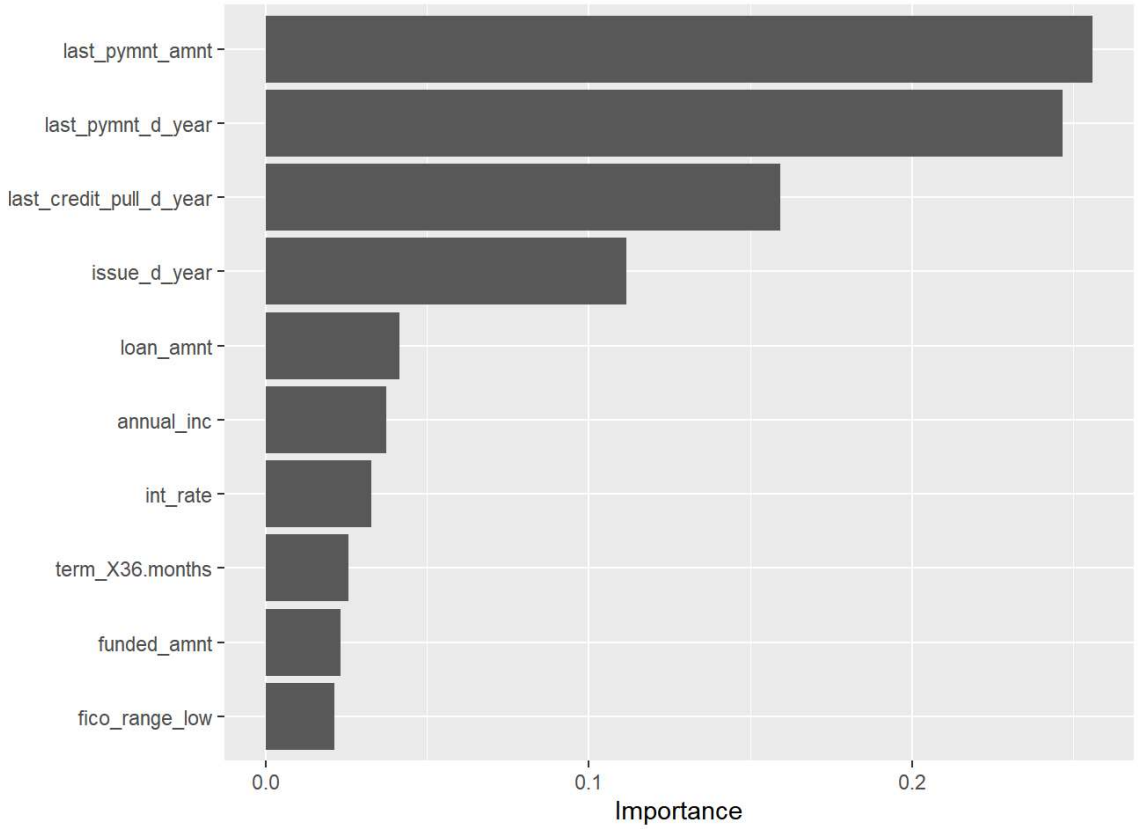
Evaluate the XGBoost Model

```
options(yardstick.event_first = FALSE)

# Variable Importance
xgb_wflow %>%
  extract_fit_parsnip() %>%
  vi()
```

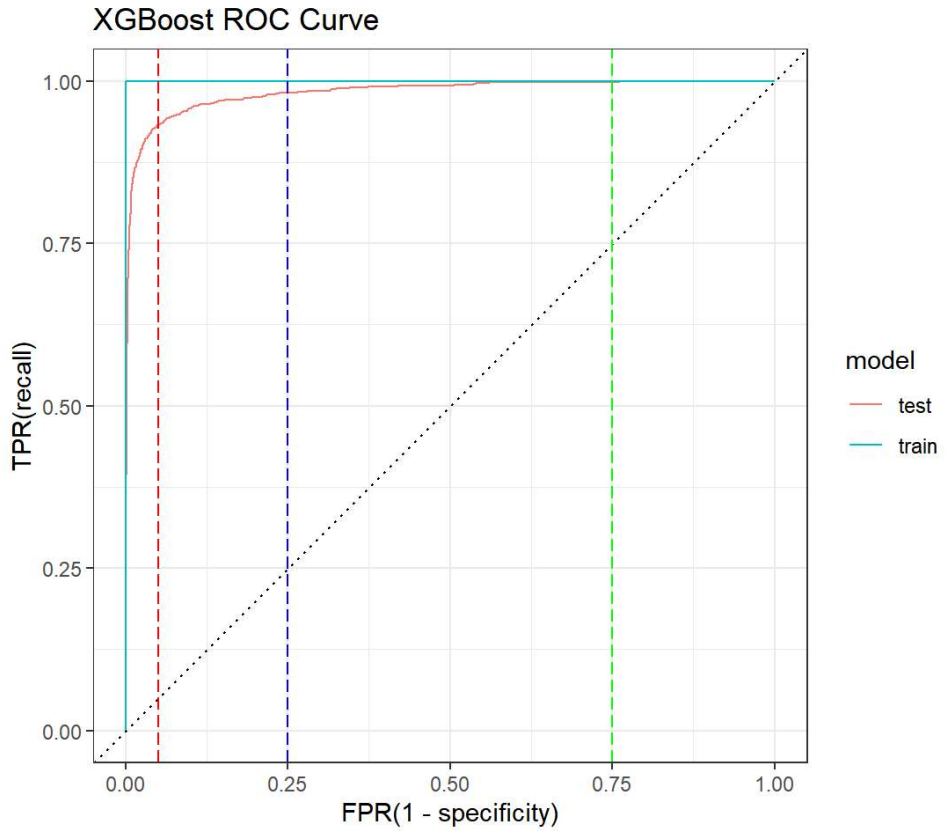
Variable <chr>	Importance <dbl>
last_pymnt_amnt	0.2558367710
last_pymnt_d_year	0.2467419861
last_credit_pull_d_year	0.1591650403
issue_d_year	0.1115855564
loan_amnt	0.0414207129
annual_inc	0.0374766467
int_rate	0.0327074081
term_X36.months	0.0258167089
funded_amnt	0.0233020884
fico_range_low	0.0213966616

1-10 of 30 rows Previous **1** 2 3 Next

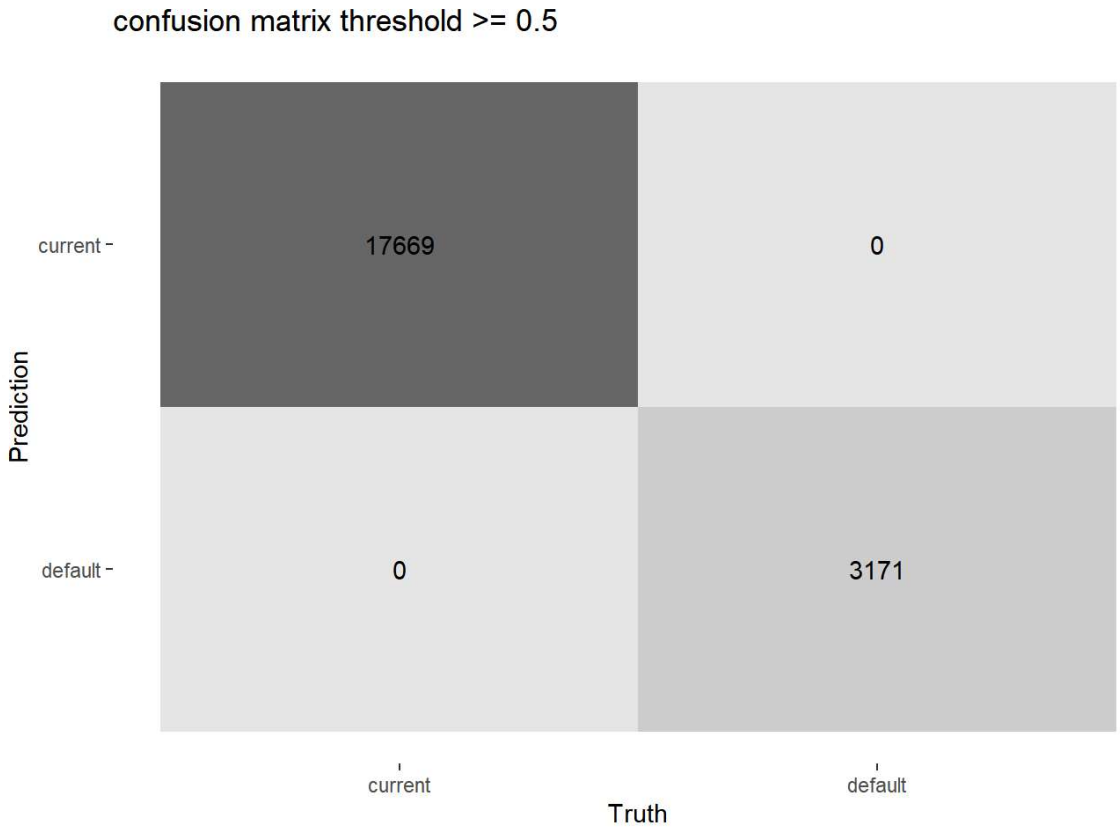
```
xgb_wflow %>%
  extract_fit_parsnip() %>%
  vip()
```



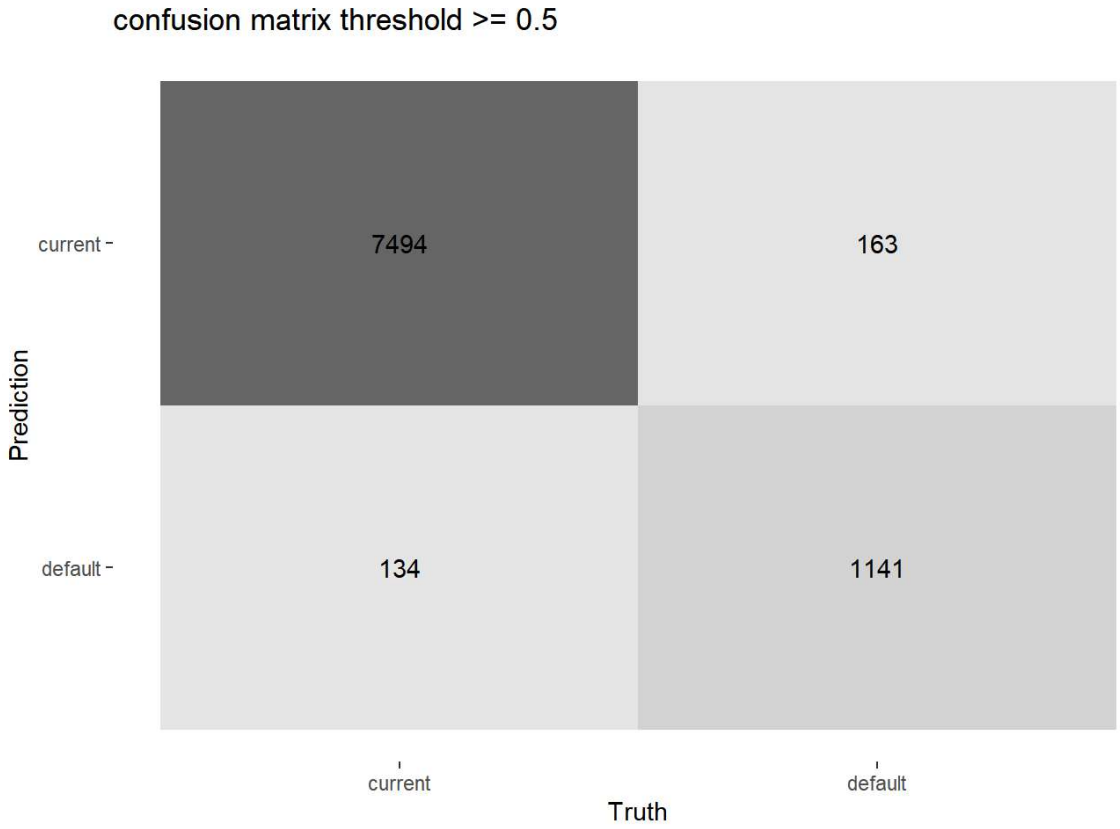

```
scored_train_xgb <- scored_train_xgb %>% mutate(part = "training")

scored_test_xgb <- scored_test_xgb %>% mutate(part = "testing")

scored_train_xgb %>% mutate(model = "train") %>%
  bind_rows(scored_test_xgb %>% mutate(model="test")) %>%
  group_by(model) %>%
  roc_curve(loan_status, .pred_default) %>%
  autoplot() +
  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 = "XGBoost ROC Curve" , x = "FPR(1 - specificity)", y = "TPR(recall)")
```




```
scored_train_xgb %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current'))) %>%
   conf_mat(loan_status, estimate = predict_class) %>%
   autoplot(type = "heatmap") +
   labs(title="confusion matrix threshold >= 0.5")
```



```
scored_test_xgb %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current'))) %>%
   conf_mat(loan_status, estimate = predict_class) %>%
   autoplot(type = "heatmap") +
   labs(title="confusion matrix threshold >= 0.5")
```



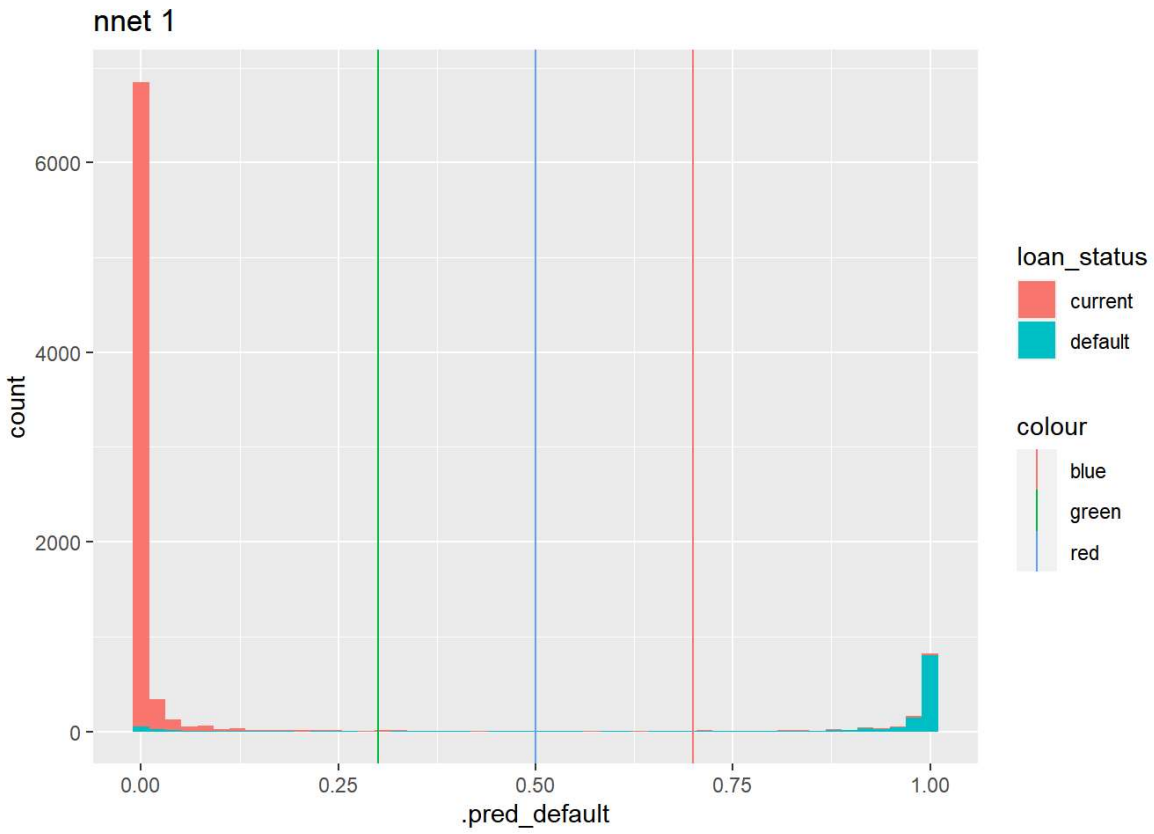
```
scored_train_xgb_mutate <- scored_train_xgb %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current')))

scored_test_xgb_mutate <- scored_test_xgb %>%
   mutate(predict_class = as.factor(if_else(.pred_default >=0.5,'default','current')))

bind_rows(scored_train_rf, scored_test_rf) %>%
    group_by(part) %>%
    dplyr::select(loan_status, .pred_class) -> train_test

precision <- train_test %>%
  yardstick::precision(loan_status, .pred_class)

recall <- train_test %>%
  yardstick::recall(loan_status, .pred_class)

scored_train_xgb %>%
    metrics(loan_status, .pred_default, estimate = .pred_class) %>%
    bind_rows(scored_train_rf_mutate %>% yardstick::precision(loan_status, predict_class)) %>%
    bind_rows(scored_train_rf_mutate %>% yardstick::recall(loan_status, predict_class)) %>%
    filter(.metric %in% c("accuracy", "roc_auc", "mn_log_loss", "precision", "recall")) %>%
    mutate(part="training") %>%
bind_rows(scored_test_xgb %>%
    metrics(loan_status, .pred_default, estimate = .pred_class) %>%
    bind_rows(scored_train_rf_mutate %>% yardstick::precision(loan_status, predict_class)) %>%
    bind_rows(scored_train_rf_mutate %>% yardstick::recall(loan_status, predict_class)) %>%
    filter(.metric %in% c("accuracy", "roc_auc", "mn_log_loss", "precision", "recall")) %>%
    mutate(part="testing"))  %>%
bind_rows(
  bind_cols(precision, recall) %>%
    mutate(f1_score = 2*.estimate...4*.estimate...8/(.estimate...4 + .estimate...8)) %>%
    mutate(.metric = "F1_Score") %>%
    mutate(.estimator = "binary") %>%
    dplyr::select(part...1, .metric, .estimator, f1_score) %>%
    rename(c(".estimate" = "f1_score"))
) %>%
    arrange(desc(part))
```

.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
accuracy	binary	1.000000000	training
mn_log_loss	binary	0.004986086	training
roc_auc	binary	1.000000000	training
precision	binary	0.970600951	training
recall	binary	0.707978556	training
F1_Score	binary	0.818745441	training
accuracy	binary	0.966748768	testing

.metric	.estimator	.estimate	part
<chr>	<chr>	<dbl>	<chr>
mn_log_loss	binary	0.120248885	testing
roc_auc	binary	0.982414040	testing
precision	binary	0.970600951	testing

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

```
# Score Distribution
scored_test_xgb %>%
  ggplot(aes(.pred_default, fill=loan_status)) +
  geom_histogram(bins=50) +
  geom_vline(aes(xintercept=.5, color="red")) +
  geom_vline(aes(xintercept=.3, color="green")) +
  geom_vline(aes(xintercept=.7, color="blue")) +
  labs(title = "nnet 1") -> scored_dist

print(scored_dist)
```



```
# operating range 0 - 10%
operating_range <- scored_test_xgb %>%
  roc_curve(loan_status, .pred_default)  %>%
  mutate(
    fpr = round((1 - specificity), 3),
    tpr = round(sensitivity, 3),
    score_threshold =  round(.threshold, 5)
  ) %>%
  group_by(fpr) %>%
  summarise(threshold = round(mean(score_threshold),4),
            tpr = mean(tpr)) %>%
  filter(fpr <= 0.5)

  print(operating_range)
```

```
## # A tibble: 501 × 3
##      fpr threshold   tpr
##    <dbl>     <dbl> <dbl>
##  1 0       Inf     0.157
##  2 0.001     0.998 0.406
##  3 0.002     0.994 0.556
##  4 0.003     0.987 0.635
##  5 0.004     0.980 0.684
##  6 0.005     0.968 0.727
##  7 0.006     0.948 0.763
##  8 0.007     0.928 0.784
##  9 0.008     0.910 0.803
## 10 0.009     0.873 0.825
## # … with 491 more rows
```

```
# Precision Recall Chart
scored_test_xgb %>%
  pr_curve(loan_status, .pred_default) %>%
  mutate(
    recall = round(recall, 2),
    .threshold = round(.threshold, 3),
    precision = round(precision, 3)
  ) %>%
  group_by(recall) %>%
  summarise(precision = max(precision),
            .threshold = min(.threshold))
```

recall	precision	.threshold
<dbl>	<dbl>	<dbl>
0.00	1.000	1.000
0.01	1.000	1.000
0.02	1.000	1.000
0.03	1.000	1.000

recall	precision	.threshold
<dbl>	<dbl>	<dbl>
0.04	1.000	1.000
0.05	1.000	1.000
0.06	0.988	1.000
0.07	0.990	1.000
0.08	0.991	1.000
0.09	0.992	1.000

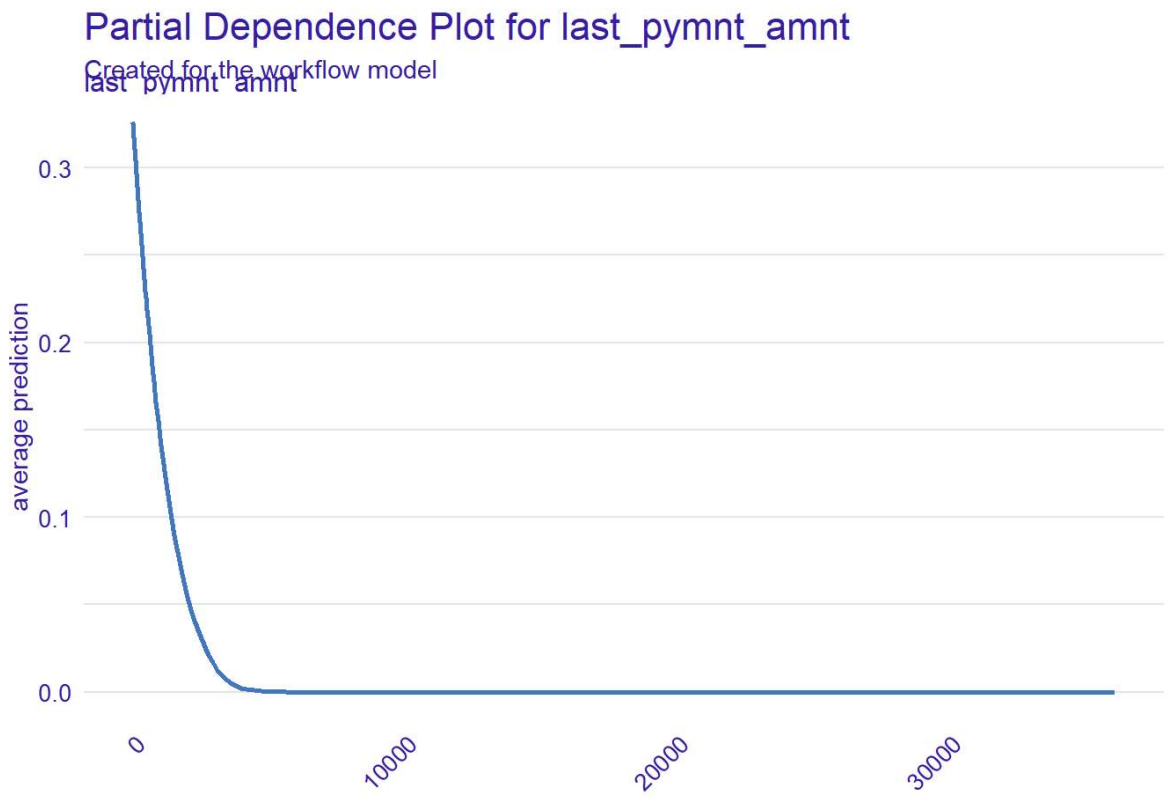
1-10 of 101 rows Previous 1 2 3 4 5 6 ... 11 Next

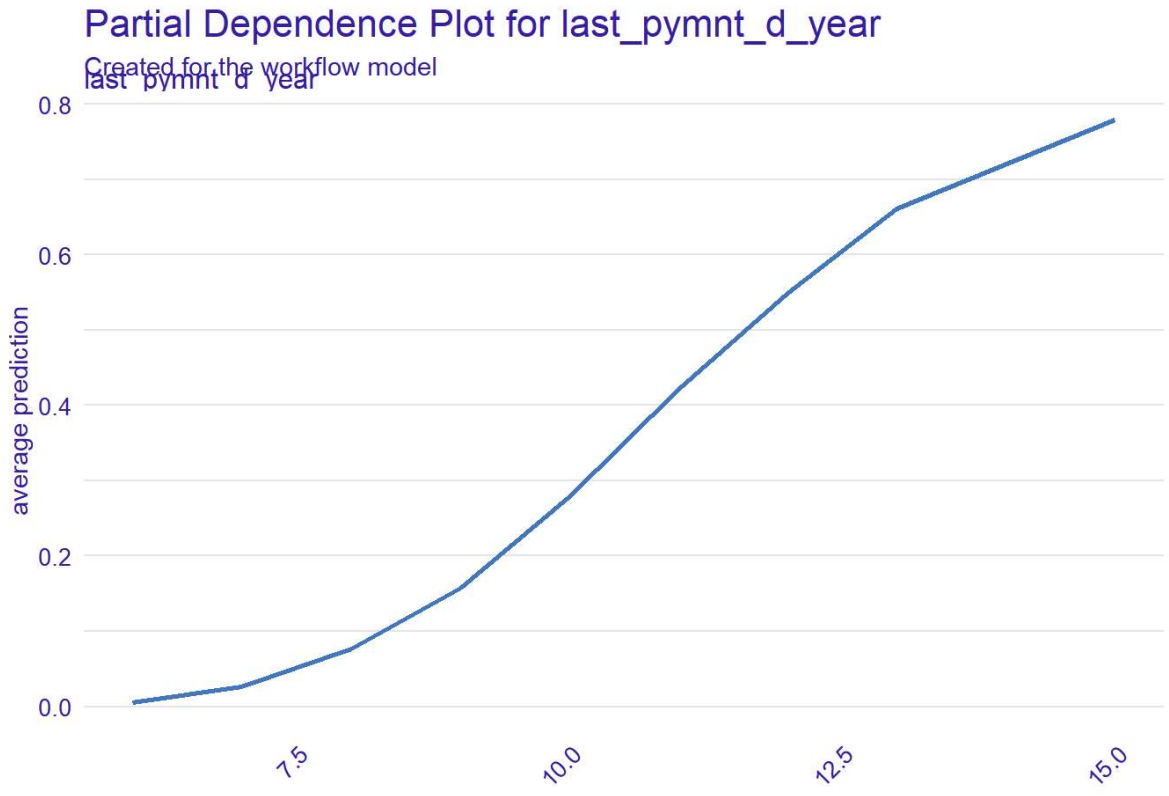
Partial Dependance Plot (XGBoost)

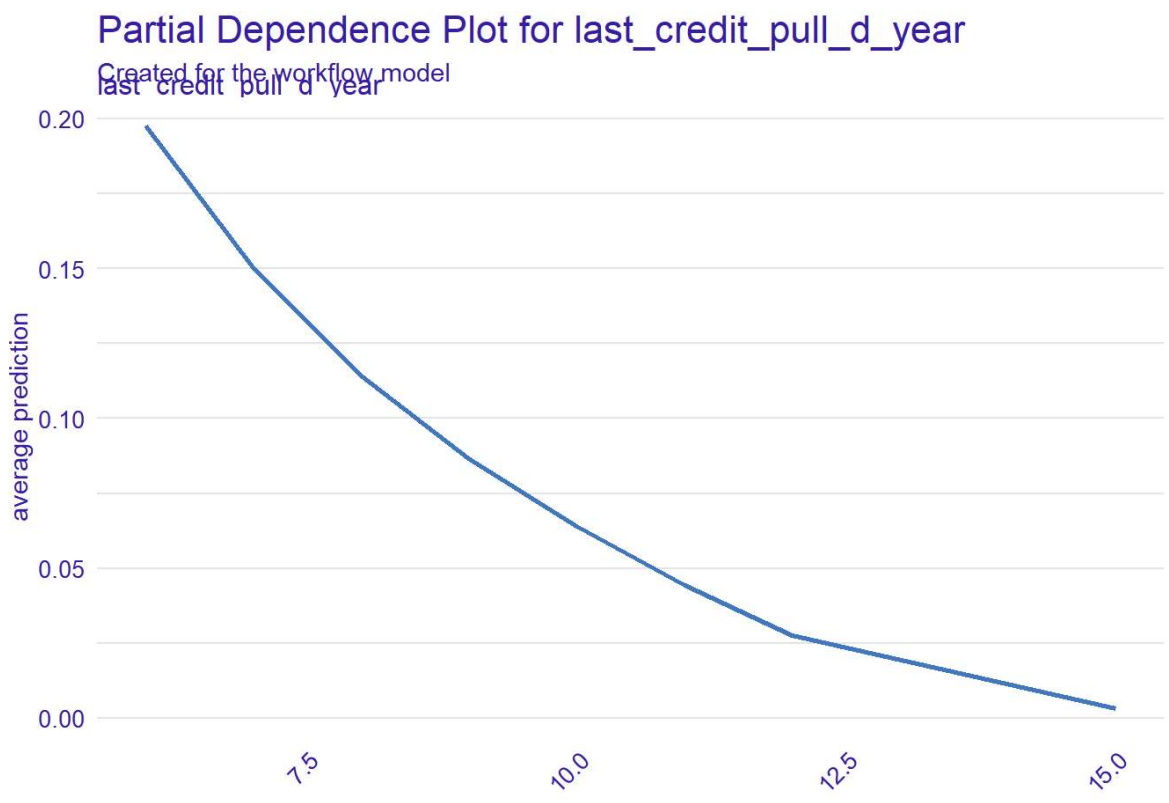
```
# create an explainer of a model


xgb_explainer <- explain_tidymodels(
  logistic_wf,
  data = train ,
  y = train$loan_default ,
  verbose = TRUE
)
```

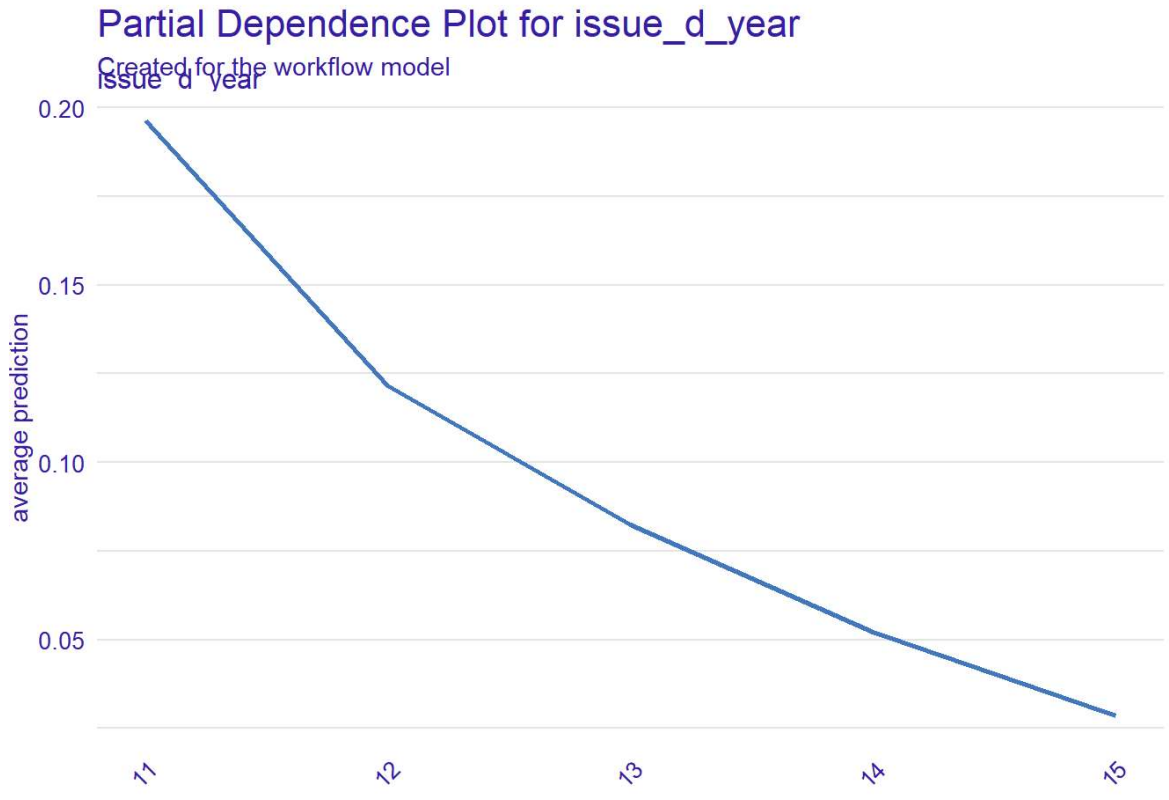
```
## Preparation of a new explainer is initiated
##   -> model label       :  workflow  (  default  )
##   -> data              :  20840  rows  32  cols
##   -> data              :  tibble converted into a data.frame
##   -> target variable   :  not specified! (  WARNING  )
##   -> predict function  :  yhat.workflow  will be used (  default  )
##   -> predicted values  :  No value for predict function target column. (  default  )
##   -> model_info        :  package tidymodels , ver. 1.0.0 , task classification (  default  )
##   -> model_info        :  Model info detected classification task but 'y' is a NULL .  (  WARNING  )
##   -> model_info        :  By deafult classification tasks supports only numercical 'y' parameter.
##   -> model_info        :  Consider changing to numerical vector with 0 and 1 values.
##   -> model_info        :  Otherwise I will not be able to calculate residuals or loss function.
##   -> predicted values  :  numerical, min =  0.00000000000000002220446 , mean =  0.1518757 , max =  0.9999777
##   -> residual function :  difference between y and yhat (  default  )
##   A new explainer has been created!
```

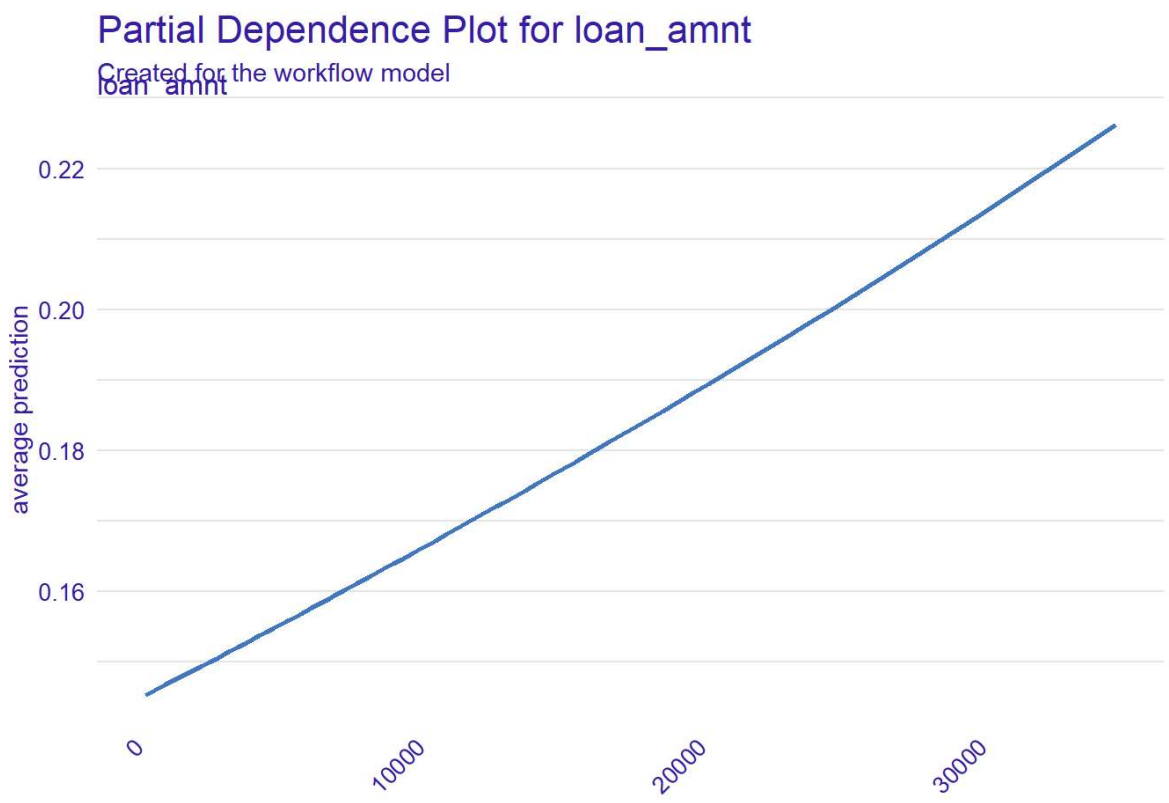
```
xgb_variable <- c('last_pymnt_amnt', 'last_pymnt_d_year', 'last_credit_pull_d_year', 'issue_d_year', 'loan_amnt')

pdp <- function(m){

# create a profile of a single variable for a model

pdp_variable <- model_profile(
  xgb_explainer,
  variables = as.character(m)
)

# Plot it

plot(pdp_variable) +
  ggtitle(paste0("Partial Dependence Plot for ", as.character(m))) +
  theme(axis.text.x=element_text(angle=45, hjust=1))

}

for (c in xgb_variable){
    print(pdp(c))
}
```


Partial Dependence Plot for last_pymnt_amnt
Created for the workflow model
last_pymnt_amnt
average prediction
0.3
0.2
0.1
0.0
0
10000
20000
30000


Partial Dependence Plot for last_pymnt_d_year
Created for the workflow model
last_pymnt_d_year
average prediction
0.8
0.6
0.4
0.2
0.0
7.5
10.0
12.5
15.0


Partial Dependence Plot for last_credit_pull_d_year
Created for the workflow model
last_credit_pull_d_year
average prediction
0.20
0.15
0.10
0.05
0.00
7.5
10.0
12.5
15.0

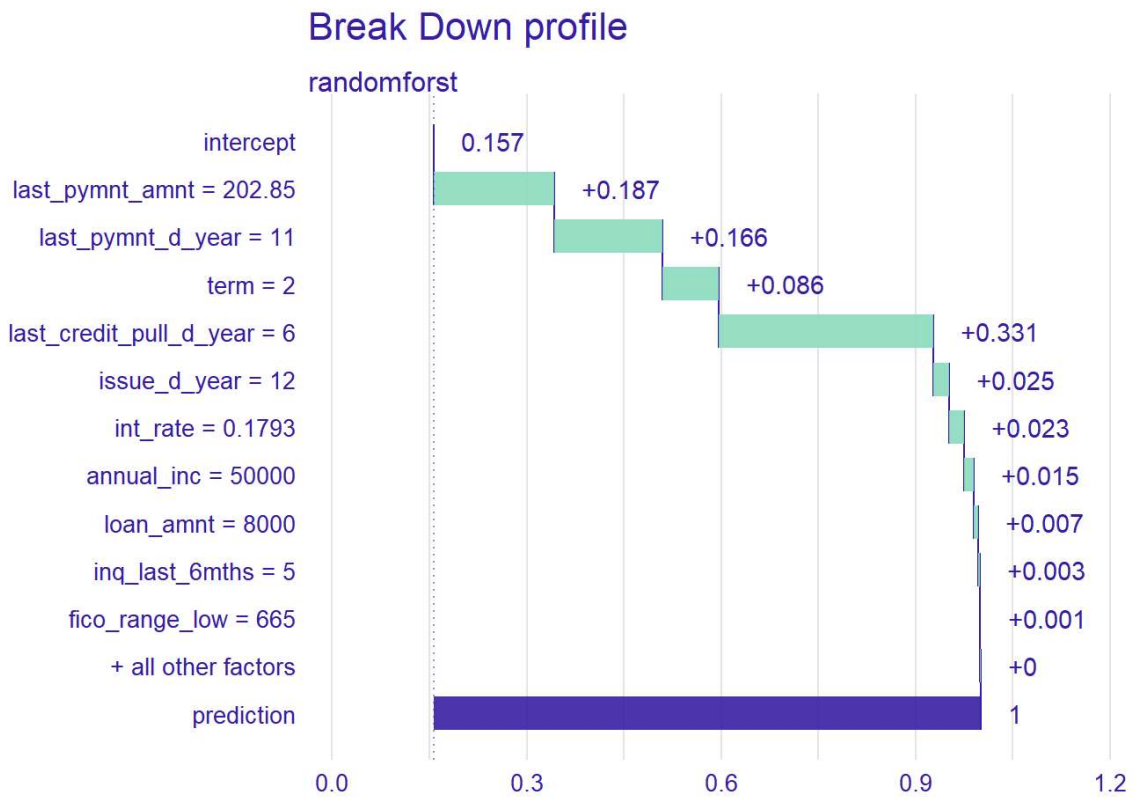

Partial Dependence Plot for issue_d_year
Created for the workflow model
issue_d_year
average prediction
0.20
0.15
0.10
0.05
11
12
13
14
15



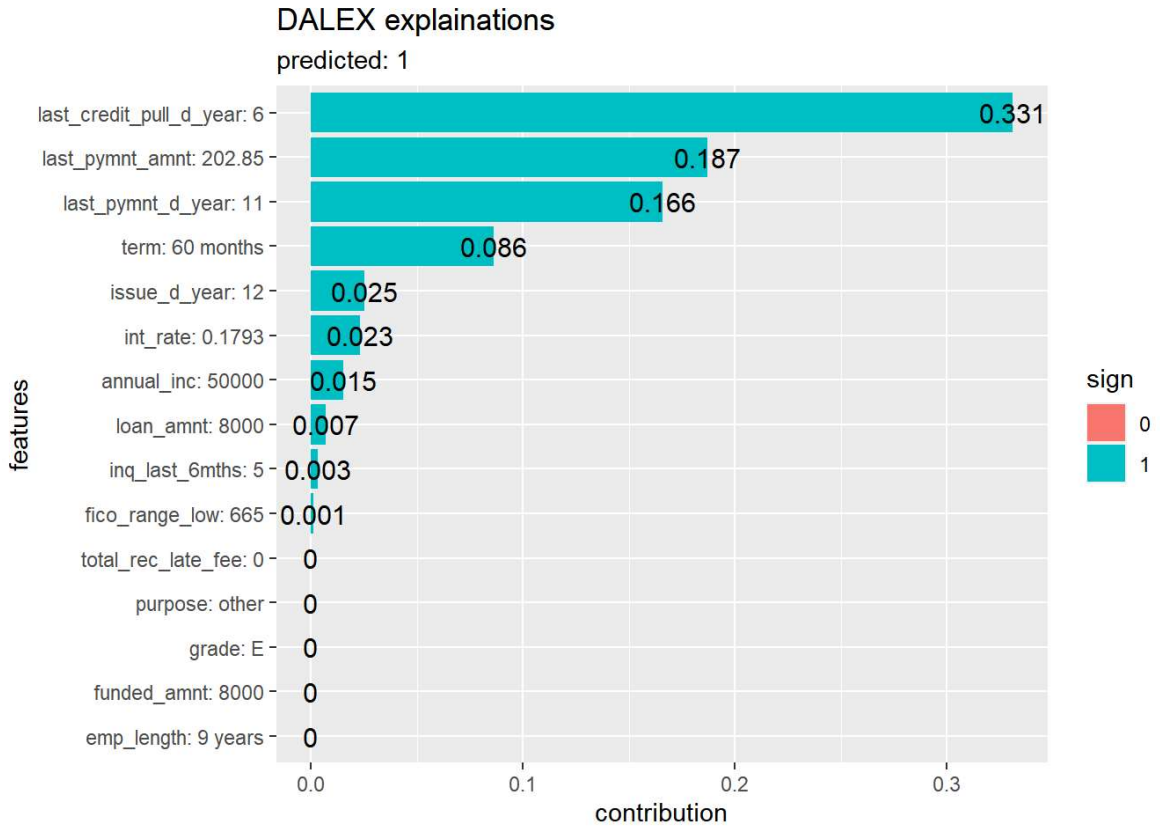
Global explanations

BREAKDOWN Explainer

```
# your model variables of interest

model_variables = c('.pred_default', 'loan_amnt', 'annual_inc', 'inq_last_6mths', 'last_pymnt_amnt', 'term', 'int_rate', 'emp
          _length', 'purpose', 'issue_d_year', 'last_pymnt_d_year', 'grade', 'last_credit_pull_d_year', 'funded_amnt', 'fico_
          range_low', 'total_rec_late_fee')


# step 1. create explainer
xgb_explainer <-
  explain_tidymodels(
    xgb_wflow,   # fitted workflow object
#    data = train,
    data = train %>% sample_n(1000),    # original training data
    y = train$loan_status, # predicted outcome
    label = "randomforst",
    verbose = FALSE
  )


# step 2. get the record you want to predict
single_record <- scored_test_xgb %>% dplyr::select(model_variables) %>%
  mutate(intercept = "", prediction = .pred_default) %>%
  slice_max(order_by = .pred_default, n=10) %>% head(1)



# step 3. run the explainer
xgb_breakdown <- predict_parts(explainer = xgb_explainer,
                               new_observation = single_record
                               )


# step 4. plot it.
# you notice you don't get categorical values ...
xgb_breakdown %>% plot()
```


Break Down profile
randomforst
intercept
last_pymnt_amnt = 202.85
last_pymnt_d_year = 11
term = 2
last_credit_pull_d_year = 6
issue_d_year = 12
int_rate = 0.1793
annual_inc = 50000
loan_amnt = 8000
inq_last_6mths = 5
fico_range_low = 665
+ all other factors
prediction
0.157
+0.187
+0.166
+0.086
+0.331
+0.025
+0.023
+0.015
+0.007
+0.003
+0.001
+0
1
0.0
0.3
0.6
0.9
1.2

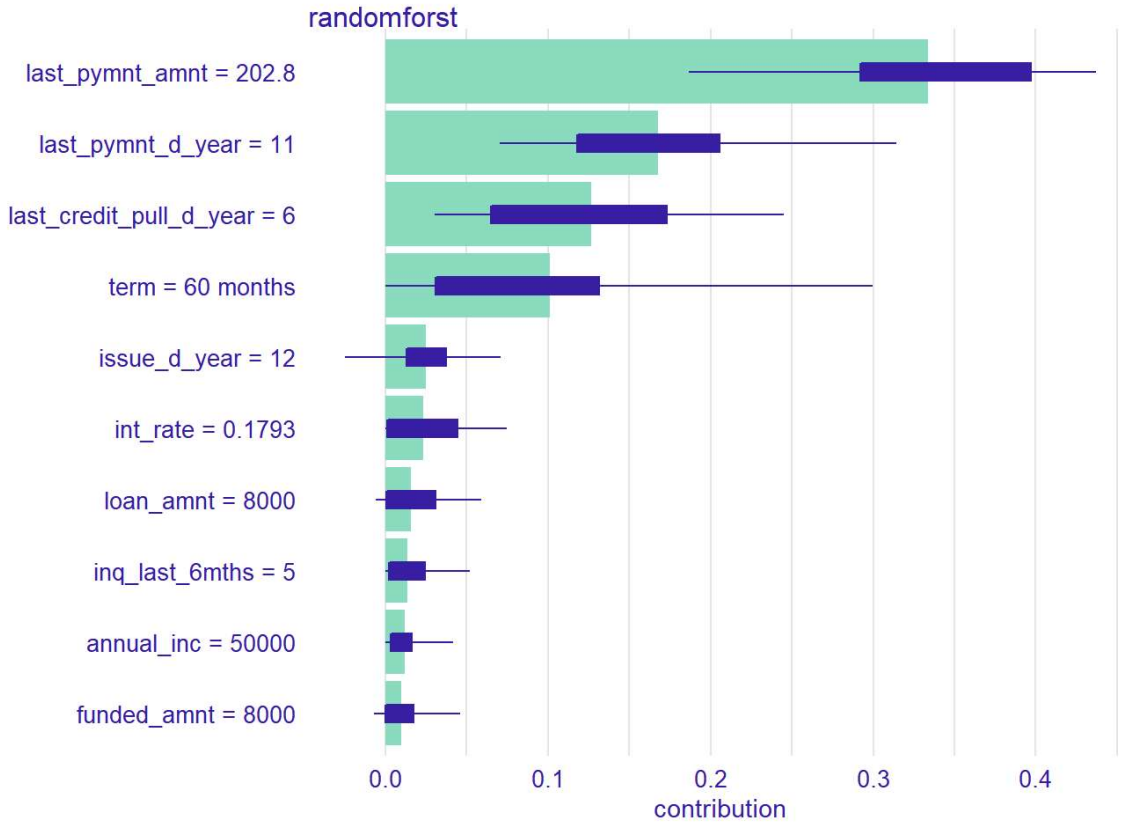
```
# --- more involved explanations with categories. ----

# step 4a.. convert breakdown to a tibble so we can join it
xgb_breakdown %>%
  as_tibble() -> breakdown_data

# step 4b. transpose your single record prediction
single_record %>%
 gather(key="variable_name",value="value") -> prediction_data

# step 4c. get a predicted probability for plot
prediction_prob <- single_record[,".pred_default"] %>% pull()

# step 5. plot it.
breakdown_data %>%
  inner_join(prediction_data) %>%
  mutate(contribution = round(contribution,3),) %>%
  filter(variable_name != "intercept") %>%
  mutate(variable = paste(variable_name,value,sep = ": ")) %>%
  ggplot(aes(y=reorder(variable, contribution), x= contribution, fill=sign)) +
  geom_col() +
  geom_text(aes(label=contribution),
          size=4,
            position=position_dodge(width=0.7),
            vjust=0.5,
            )+
  labs(
    title = "DALEX explainations",
    subtitle = paste("predicted:",as.character(round(prediction_prob,3))),
                    x="contribution",
                    y="features")
```



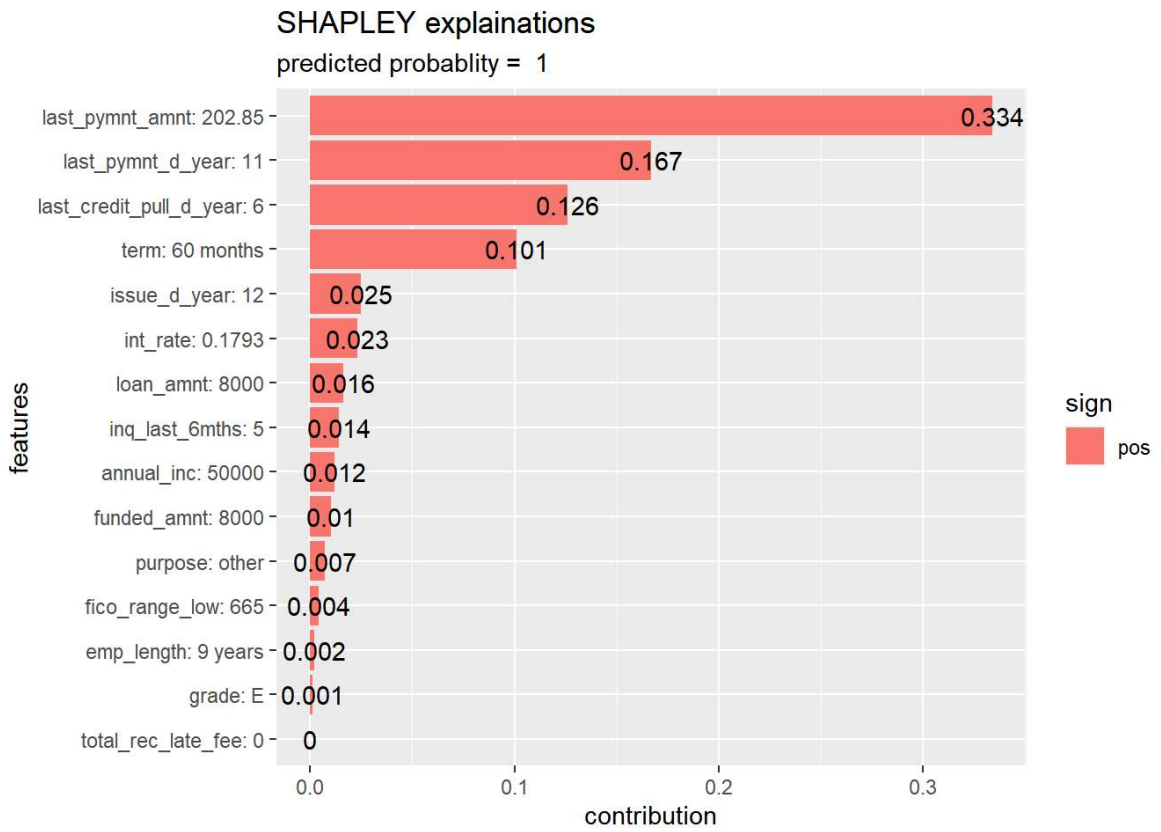
Local Explanations

SHAPLEY Explainer

```
# step 3. run the explainer
xgb_shapley <- predict_parts(explainer = xgb_explainer,
                             new_observation = single_record,
                             type="shap")

# step 4. plot it.
# you notice you don't get categorical values ...
xgb_shapley %>% plot()
```




```
# --- more involved explanations with categories. ----

# step 4a.. convert breakdown to a tibble so we can join it
xgb_shapley %>%
  as_tibble() -> shap_data

# step 4b. transpose your single record prediction
single_record %>%
 gather(key="variable_name",value="value") -> prediction_data

# step 4c. get a predicted probability for plot
prediction_prob <- single_record[,".pred_default"] %>% mutate(.pred_default = round(.pred_default,3)) %>% pull()

# step 5. plot it.
shap_data %>%
  inner_join(prediction_data) %>%
  mutate(variable = paste(variable_name,value,sep = ": ")) %>%
  group_by(variable) %>%
  summarize(contribution = mean(contribution)) %>%
  mutate(contribution = round(contribution,3),
         sign = if_else(contribution < 0, "neg","pos")) %>%
  ggplot(aes(y=reorder(variable, contribution), x= contribution, fill=sign)) +
  geom_col() +
  geom_text(aes(label=contribution))+
  labs(
    title = "SHAPLEY explainations",
    subtitle = paste("predicted probablity = ",prediction_prob) ,
                  x="contribution",
                  y="features")
```


SHAPLEY explainations
predicted probablity = 1
last_pymnt_amnt: 202.85
last_pymnt_d_year: 11
last_credit_pull_d_year: 6
term: 60 months
issue_d_year: 12
int_rate: 0.1793
loan_amnt: 8000
inq_last_6mths: 5
annual_inc: 50000
funded_amnt: 8000
purpose: other
fico_range_low: 665
emp_length: 9 years
grade: E
total_rec_late_fee: 0
0.334
0.167
0.126
0.101
0.025
0.023
0.016
0.014
0.012
0.01
0.007
0.004
0.002
0.001
0
features
0.0
0.1
0.2
0.3
contribution
sign
pos

Make a Function

```
xgb_explainer <- explain_tidymodels(
    rf_wflow,   # fitted workflow object
    data = train %>% sample_n(1000),    # original training data
    y = train$loan_status, # predicted outcome
    label = "randomforest",
    verbose = FALSE
  )

explain_prediction <- function(single_record){
  # step 3. run the explainer
record_shap <- predict_parts(explainer = xgb_explainer,
                             new_observation = single_record,
                             type="shap")

# step 4. plot it.
# you notice you don't get categorical values ...
record_shap %>% plot() %>% print()

# --- more involved explanations with categories. ----

# step 4a.. convert breakdown to a tibble so we can join it
record_shap %>%
  as_tibble() -> shap_data

# step 4b. transpose your single record prediction
single_record %>%
 gather(key="variable_name",value="value") -> prediction_data

# step 4c. get a predicted probability for plot
prediction_prob <- single_record[,".pred_default"] %>% mutate(.pred_default = round(.pred_default,3)) %>% pull()

# step 5. plot it.
shap_data %>%
  inner_join(prediction_data) %>%
  mutate(variable = paste(variable_name,value,sep = ": ")) %>%
  group_by(variable) %>%
  summarize(contribution = mean(contribution)) %>%
  mutate(contribution = round(contribution,3),
         sign = if_else(contribution < 0, "neg","pos")) %>%
  ggplot(aes(y=reorder(variable, contribution), x= contribution, fill=sign)) +
  geom_col() +
  geom_text(aes(label=contribution))+
  labs(
    title = "SHAPLEY explainations",
    subtitle = paste("predicted probablity = ",prediction_prob) ,
                  x="contribution",
                  y="features")

}
```

```
scored_test_xgb %>%
  filter(.pred_class == 'default') %>%
  filter(.pred_class == 'default') %>%
  slice_max(.pred_default,n=10)
```

.pred_current <dbl>	.pred_default <dbl>	.pred_class <fct>	loan_status <fct>	loan_amnt <dbl>	funded_amnt <dbl>	funded_amnt_inv <dbl>	int_rate <dbl>
0.000008220754	0.9999918	default	default	8000	8000	6725.0000	0.1793
0.000010558794	0.9999894	default	default	10000	10000	9175.0000	0.1316
0.000011241410	0.9999888	default	default	10000	10000	9975.0000	0.1311
0.000013776330	0.9999862	default	default	10000	10000	9950.0000	0.1385
0.000014257276	0.9999857	default	default	5000	5000	5000.0000	0.1348
0.000017836512	0.9999822	default	default	18000	13250	13250.0000	0.2264
0.000023066857	0.9999769	default	default	15000	15000	14628.4400	0.1459
0.000027022774	0.9999730	default	default	10000	10000	9900.0000	0.1218
0.000027126385	0.9999729	default	default	20000	20000	575.0019	0.1299
0.000027843398	0.9999722	default	default	6000	6000	5950.0000	0.1632

1-10 of 10 rows | 1-8 of 36 columns

```
scored_test_xgb %>%
   filter(.pred_class == 'default') %>%
   filter(loan_status != 'default' ) %>%
  slice_max(.pred_default,n=10)
```

.pred_current <dbl>	.pred_default <dbl>	.pred_class <fct>	loan_status <fct>	loan_amnt <dbl>	funded_amnt <dbl>	funded_amnt_inv <dbl>	int_rate <dbl>
0.00009785406	0.9999021	default	current	2000	2000	2000.00	0.1316
0.00032366693	0.9996763	default	current	5000	5000	5000.00	0.0999
0.00078116619	0.9992188	default	current	12000	12000	9887.01	0.1316
0.00108001370	0.9989200	default	current	13000	13000	13000.00	0.1749
0.00110448664	0.9988955	default	current	4075	4075	4050.00	0.0832
0.00136160932	0.9986384	default	current	12000	12000	12000.00	0.1036
0.00142461131	0.9985754	default	current	20000	20000	19950.00	0.1479
0.00145760225	0.9985424	default	current	4000	4000	3750.00	0.1148
0.00201453571	0.9979855	default	current	12000	12000	12000.00	0.1599
0.00230993028	0.9976901	default	current	12000	12000	12000.00	0.1527

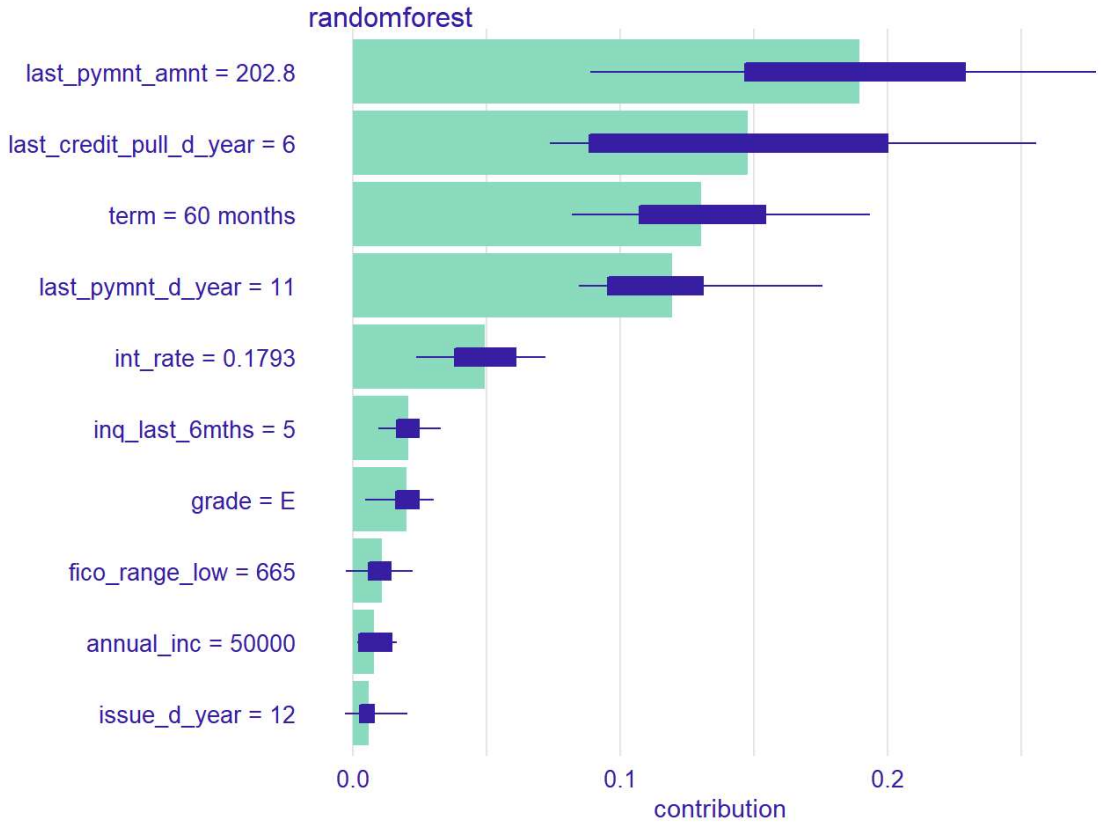
1-10 of 10 rows | 1-8 of 36 columns

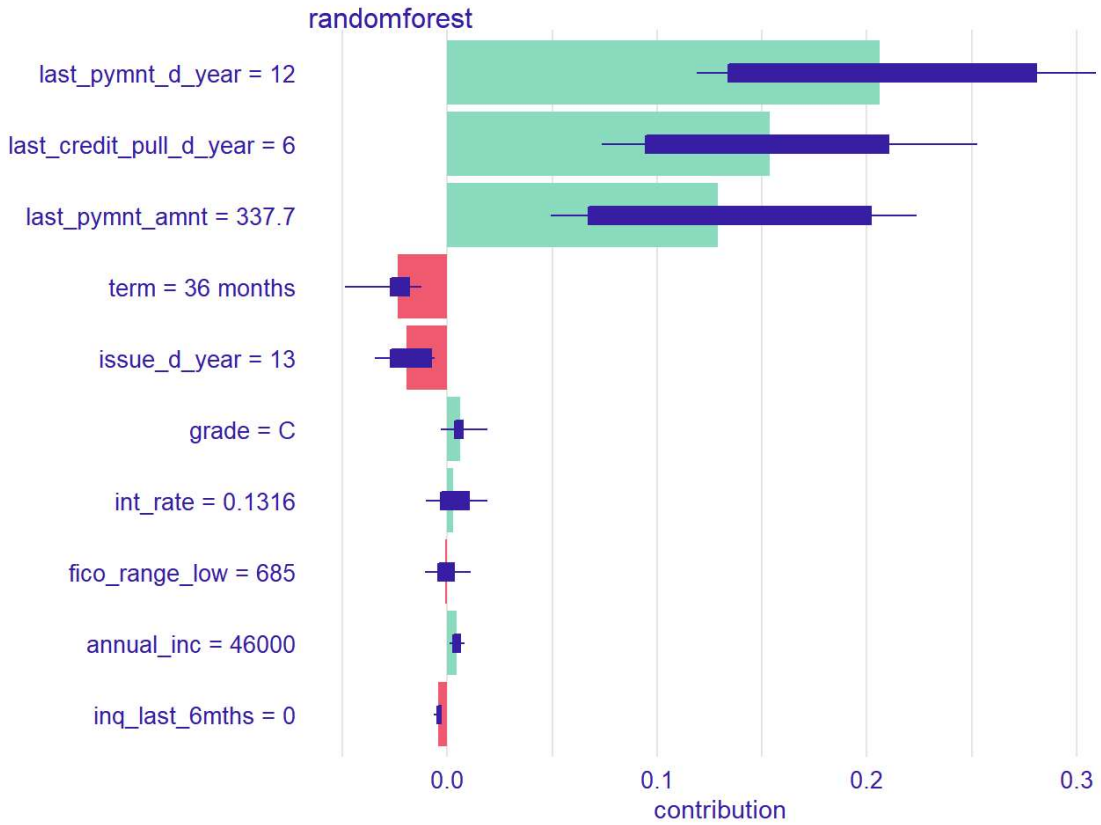
```
scored_test_xgb %>%
  filter(.pred_class != 'default') %>%
  filter(loan_status == 'default' ) %>%
  slice_min(.pred_default,n=10)
```

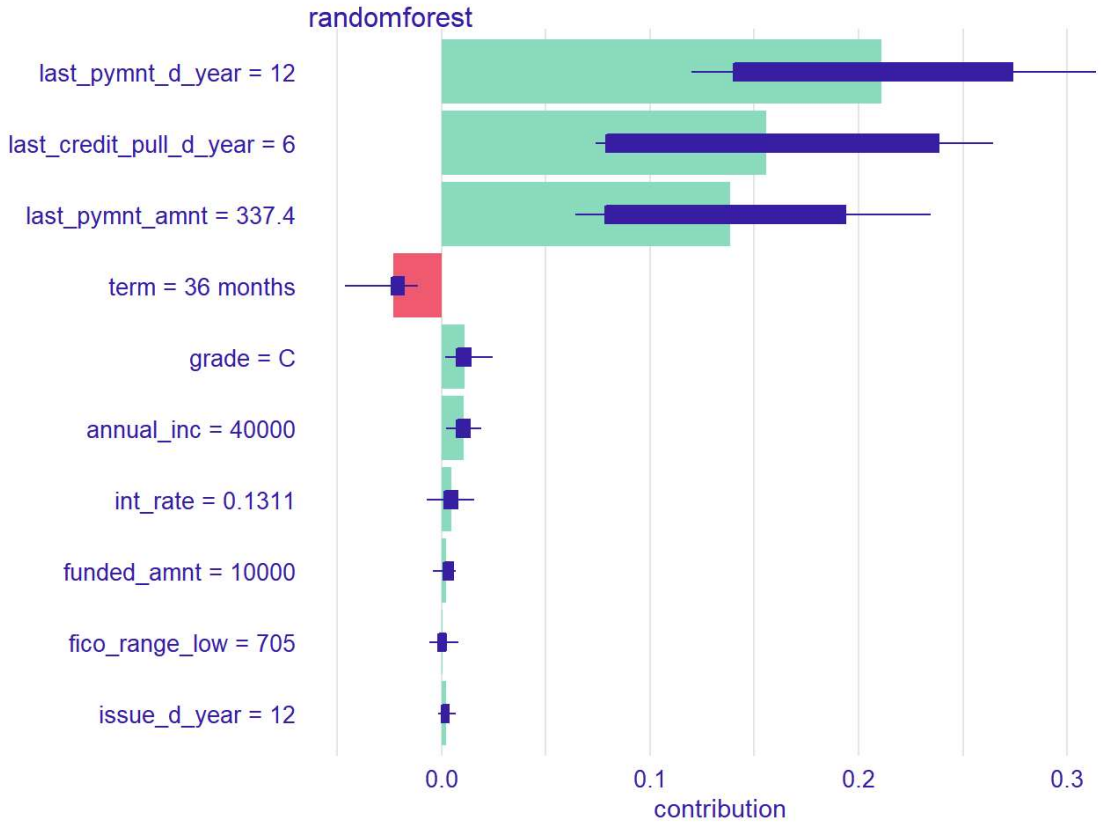
.pred_current <dbl>	.pred_default <dbl>	.pred_class <fct>	loan_status <fct>	loan_amnt <dbl>	funded_amnt <dbl>	funded_amnt_inv <dbl>	int_rate <dbl>
0.9999849	0.00001513958	current	default	3600	3600	3600.000	0.1398
0.9999803	0.00001966953	current	default	15850	15850	15844.212	0.1565
0.9999504	0.00004959106	current	default	6500	6500	6496.816	0.0579
0.9999201	0.00007987022	current	default	1000	1000	900.000	0.1148
0.9999171	0.00008285046	current	default	10000	10000	10000.000	0.1242
0.9999070	0.00009298325	current	default	25000	15925	15850.000	0.2077
0.9999046	0.00009536743	current	default	6000	6000	6000.000	0.0849
0.9998919	0.00010812283	current	default	8000	8000	8000.000	0.0890
0.9998821	0.00011789799	current	default	12250	12250	12250.000	0.0691
0.9997916	0.00020843744	current	default	4000	4000	4000.000	0.1399

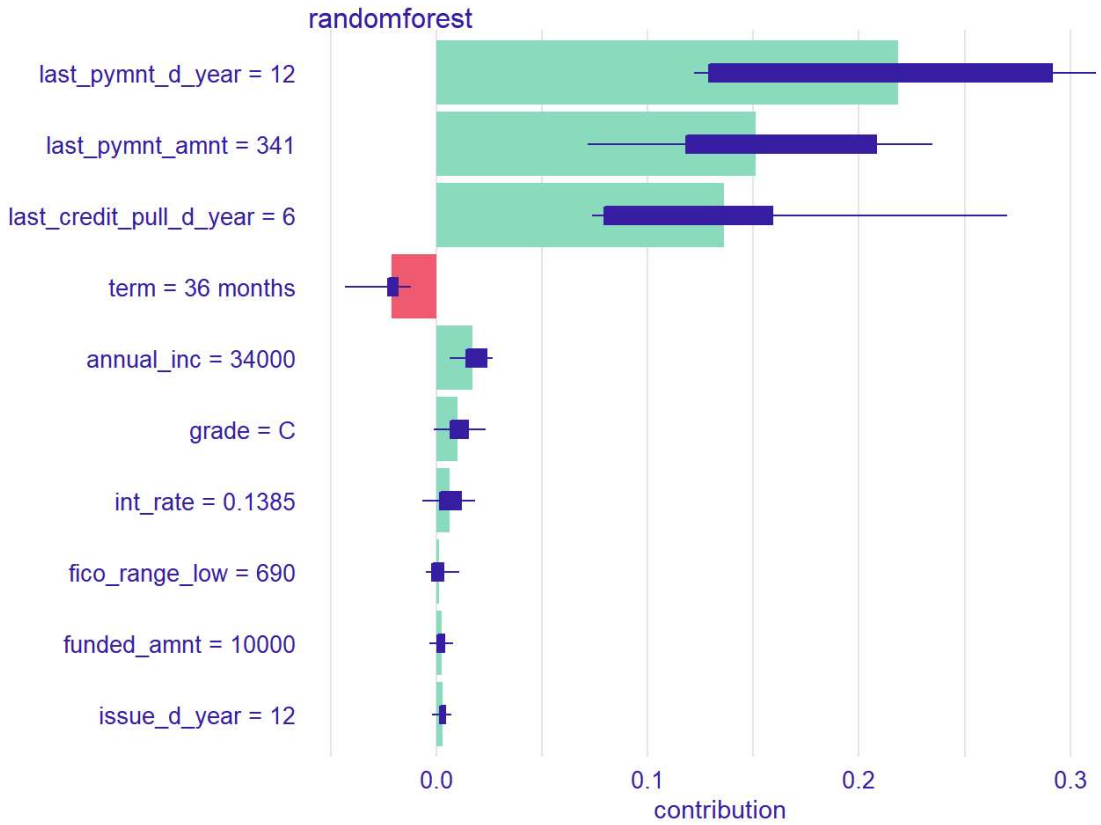
1-10 of 10 rows | 1-8 of 36 columns

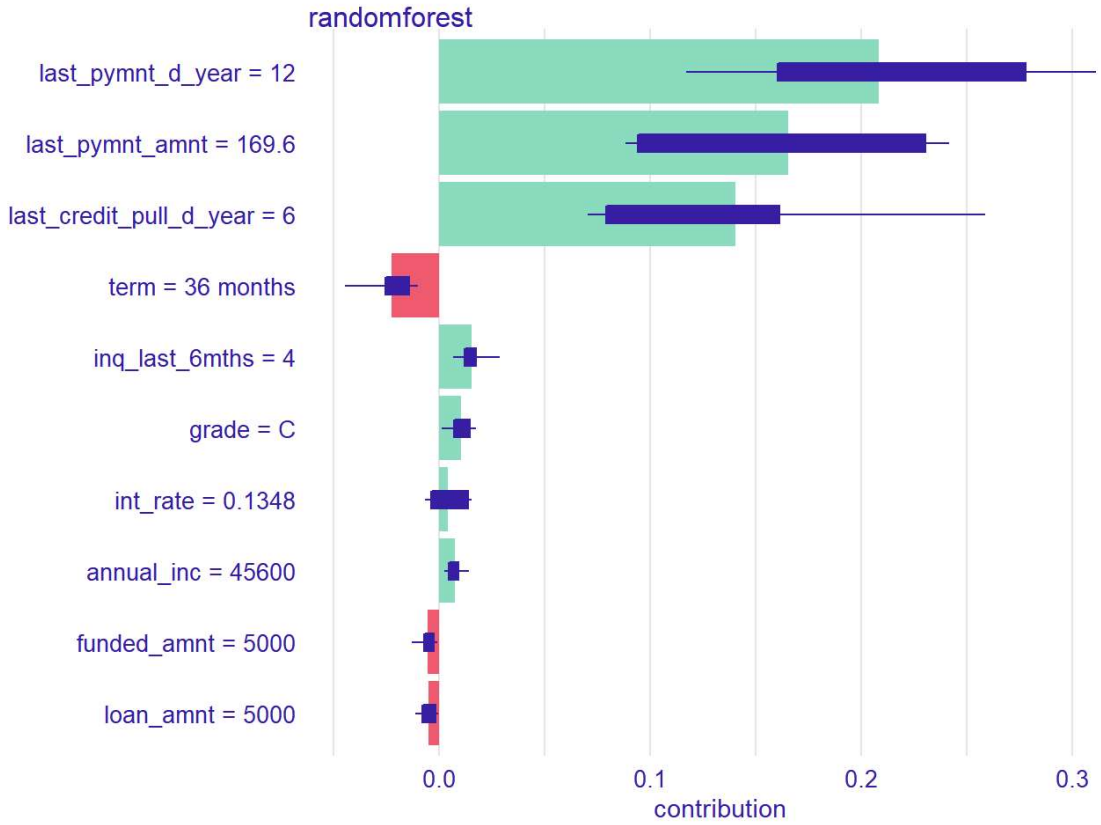
```
top_5_tp <- scored_test_xgb %>%
  filter(.pred_class == 'default') %>%
  filter(.pred_class == 'default') %>%
  slice_max(.pred_default,n=5)

top_5_fp <- scored_test_xgb %>%
   filter(.pred_class == 'default') %>%
   filter(loan_status != 'default' ) %>%
  slice_max(.pred_default,n=5)

top_5_fn <- scored_test_xgb %>%
  filter(.pred_class != 'default') %>%
  filter(loan_status == 'default' ) %>%
  slice_min(.pred_default,n=5)

# repeat for TP, FP and FN
for (row in 1:nrow(top_5_tp)) {
    s_record <- top_5_tp[row,]
    explain_prediction(s_record)
}
```


randomforest
last_pymnt_amnt = 202.8
last_credit_pull_d_year = 6
term = 60 months
last_pymnt_d_year = 11
int_rate = 0.1793
inq_last_6mths = 5
grade = E
fico_range_low = 665
annual_inc = 50000
issue_d_year = 12
0.0
0.1
0.2
contribution

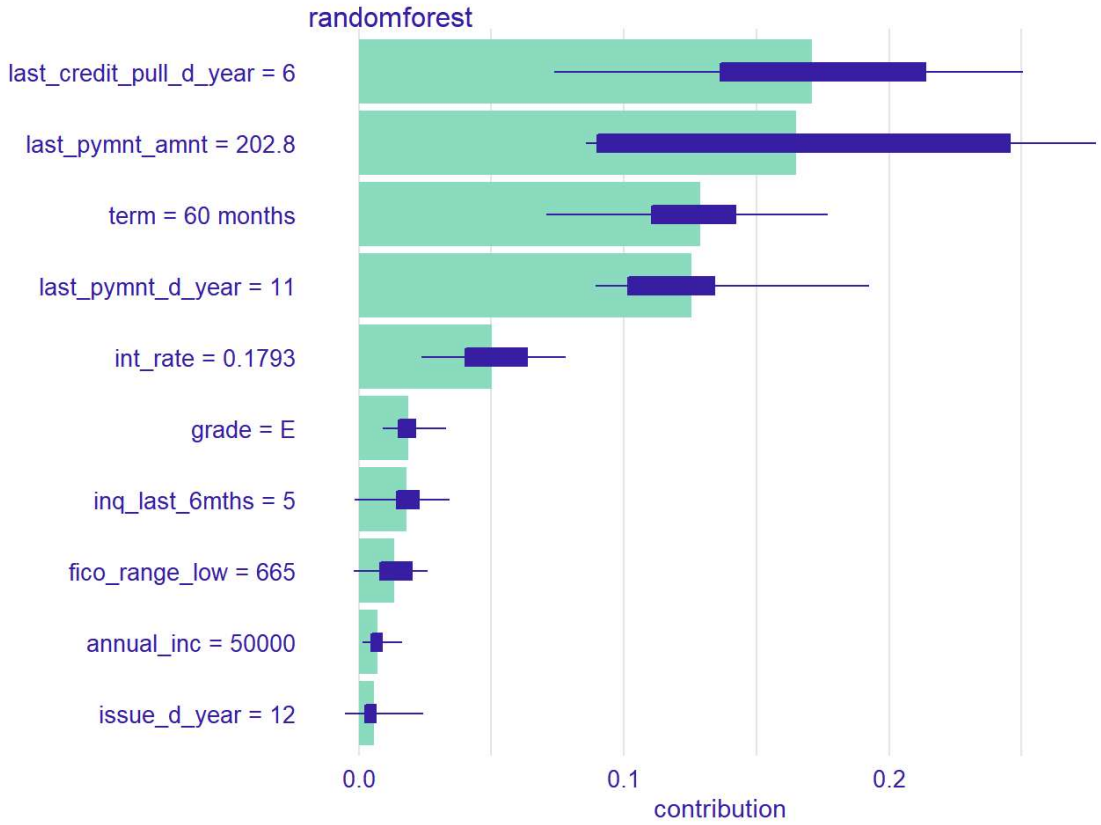

randomforest
last_pymnt_d_year = 12
last_credit_pull_d_year = 6
last_pymnt_amnt = 337.7
term = 36 months
issue_d_year = 13
grade = C
int_rate = 0.1316
fico_range_low = 685
annual_inc = 46000
inq_last_6mths = 0
0.0
0.1
0.2
0.3
contribution

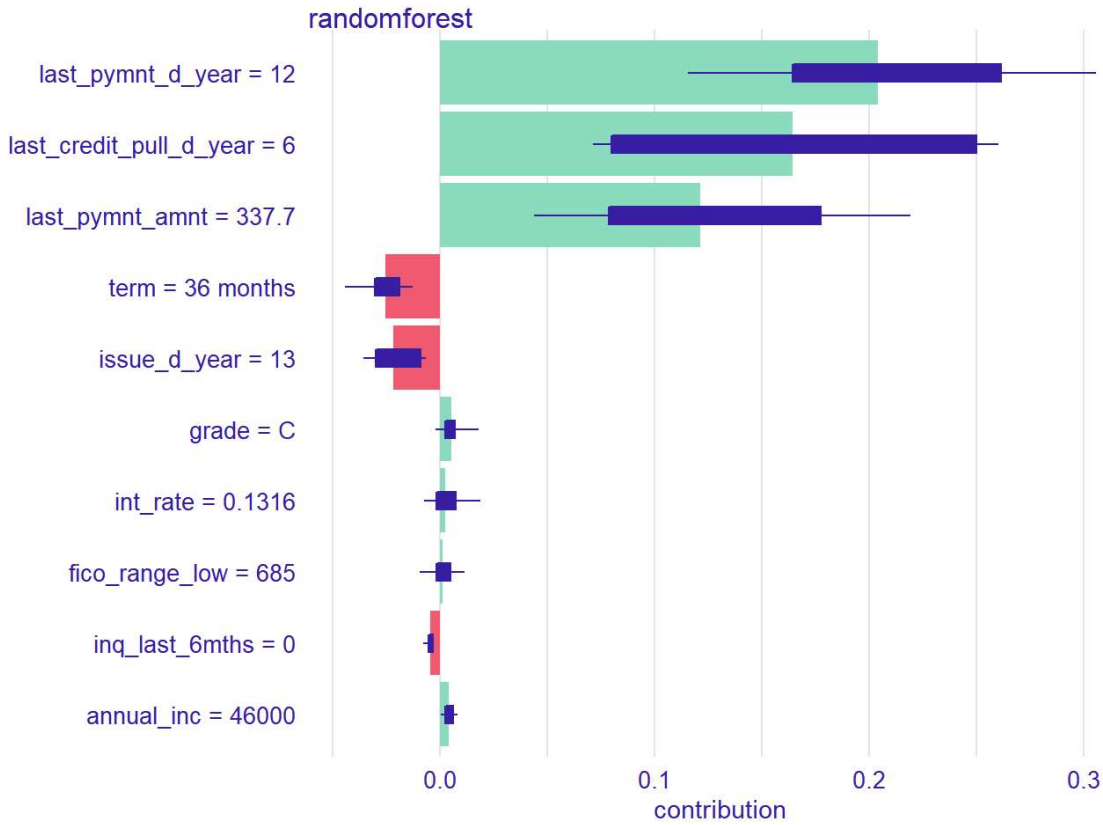

randomforest
last_pymnt_d_year = 12
last_credit_pull_d_year = 6
last_pymnt_amnt = 337.4
term = 36 months
grade = C
annual_inc = 40000
int_rate = 0.1311
funded_amnt = 10000
fico_range_low = 705
issue_d_year = 12
0.0
0.1
0.2
0.3
contribution

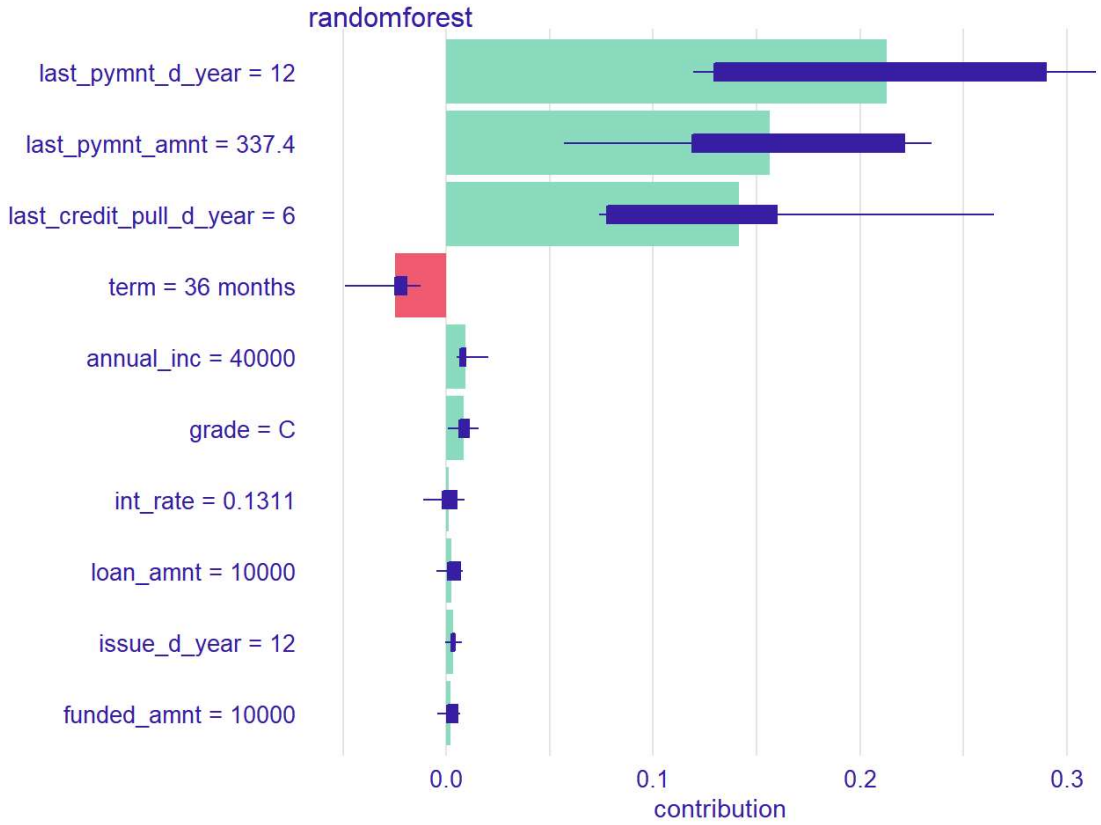

randomforest
last_pymnt_d_year = 12
last_pymnt_amnt = 341
last_credit_pull_d_year = 6
term = 36 months
annual_inc = 34000
grade = C
int_rate = 0.1385
fico_range_low = 690
funded_amnt = 10000
issue_d_year = 12
0.0
0.1
0.2
0.3
contribution

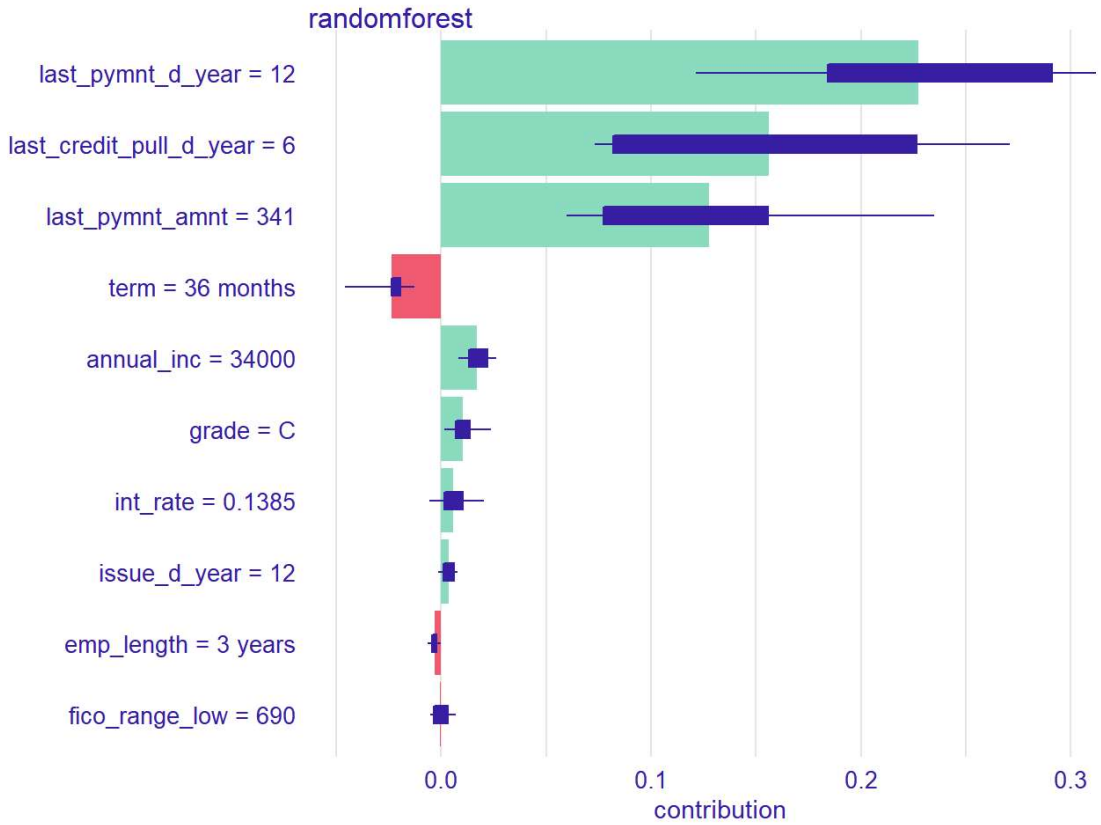


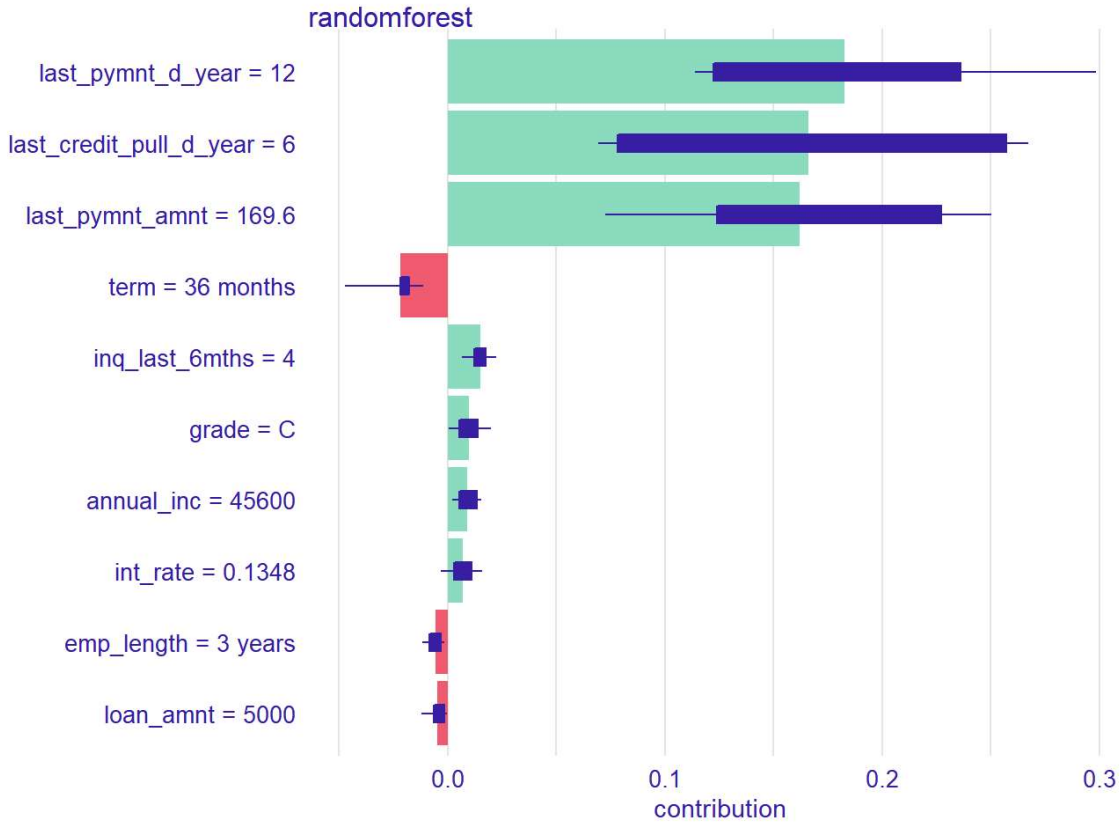

```
for (row in 1:nrow(top_5_fp)) {
    s_record <- top_5_tp[row,]
    explain_prediction(s_record)
}
```


randomforest
last_credit_pull_d_year = 6
last_pymnt_amnt = 202.8
term = 60 months
last_pymnt_d_year = 11
int_rate = 0.1793
grade = E
inq_last_6mths = 5
fico_range_low = 665
annual_inc = 50000
issue_d_year = 12
0.0
0.1
0.2
contribution

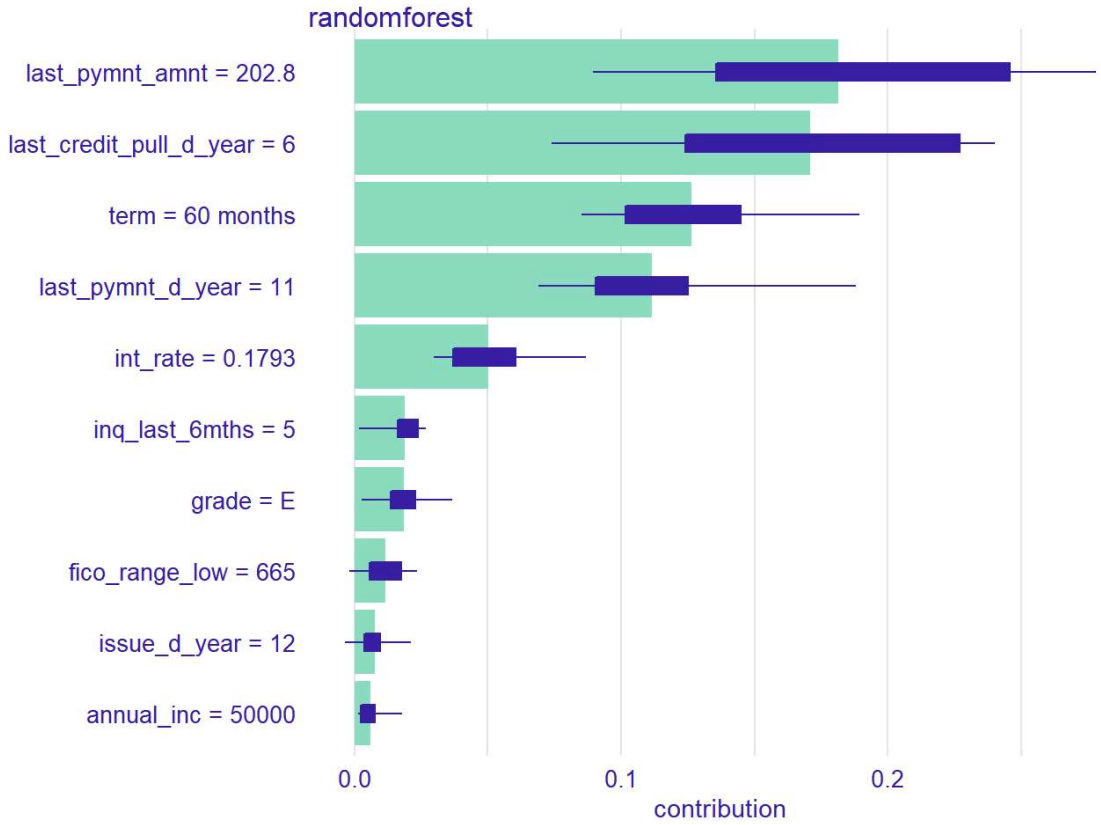

randomforest
last_pymnt_d_year = 12
last_credit_pull_d_year = 6
last_pymnt_amnt = 337.7
term = 36 months
issue_d_year = 13
grade = C
int_rate = 0.1316
fico_range_low = 685
inq_last_6mths = 0
annual_inc = 46000
0.0
0.1
0.2
0.3
contribution

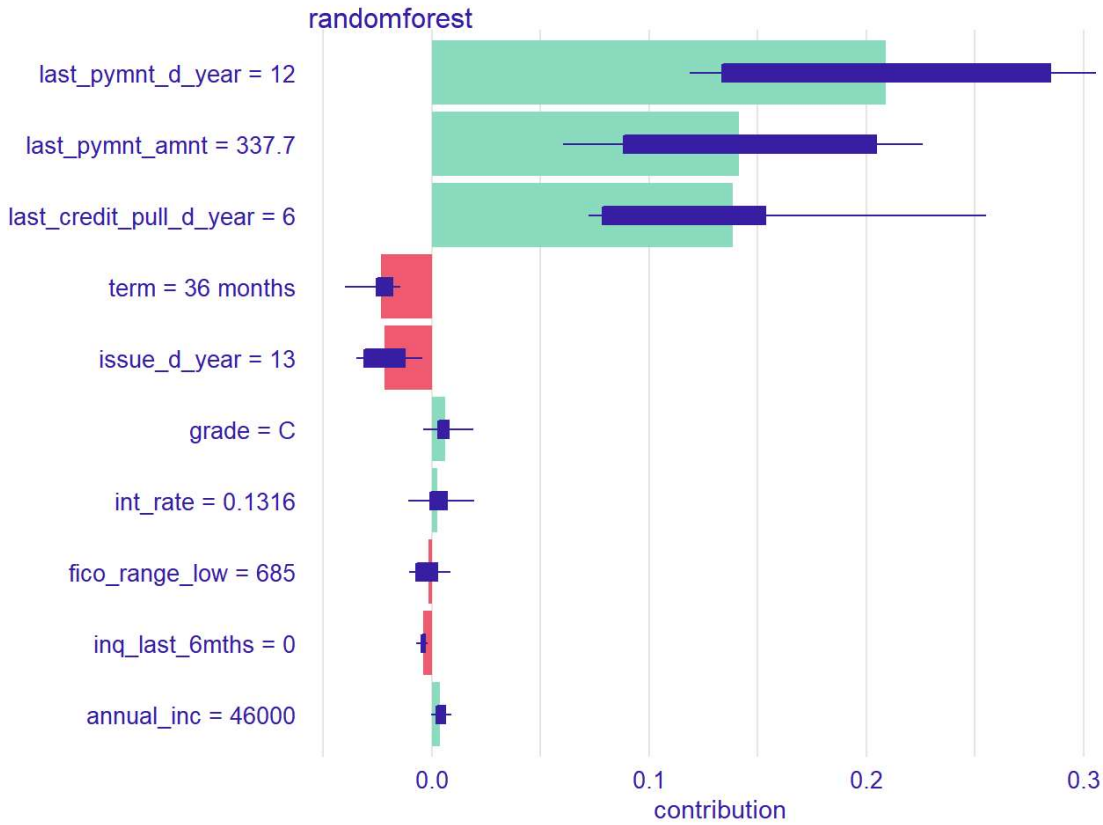

randomforest
last_pymnt_d_year = 12
last_pymnt_amnt = 337.4
last_credit_pull_d_year = 6
term = 36 months
annual_inc = 40000
grade = C
int_rate = 0.1311
loan_amnt = 10000
issue_d_year = 12
funded_amnt = 10000
0.0
0.1
0.2
0.3
contribution

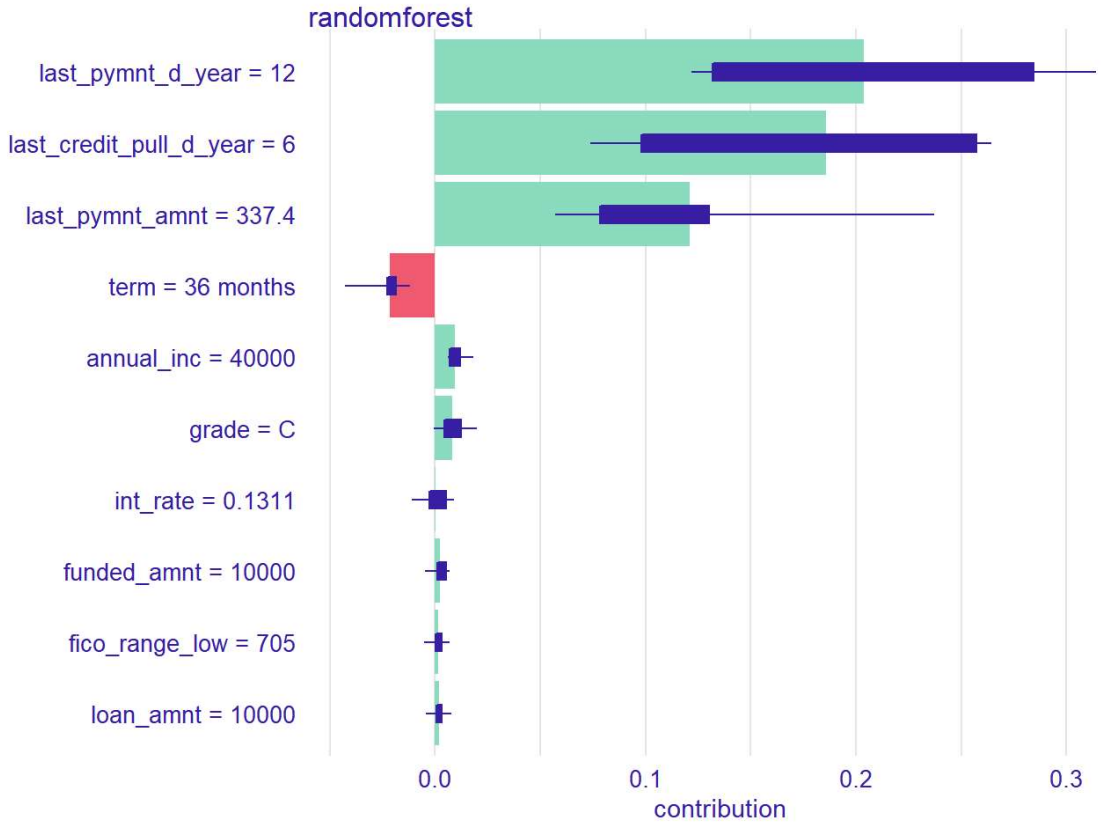

randomforest
last_pymnt_d_year = 12
last_credit_pull_d_year = 6
last_pymnt_amnt = 341
term = 36 months
annual_inc = 34000
grade = C
int_rate = 0.1385
issue_d_year = 12
emp_length = 3 years
fico_range_low = 690
0.0
0.1
0.2
0.3
contribution

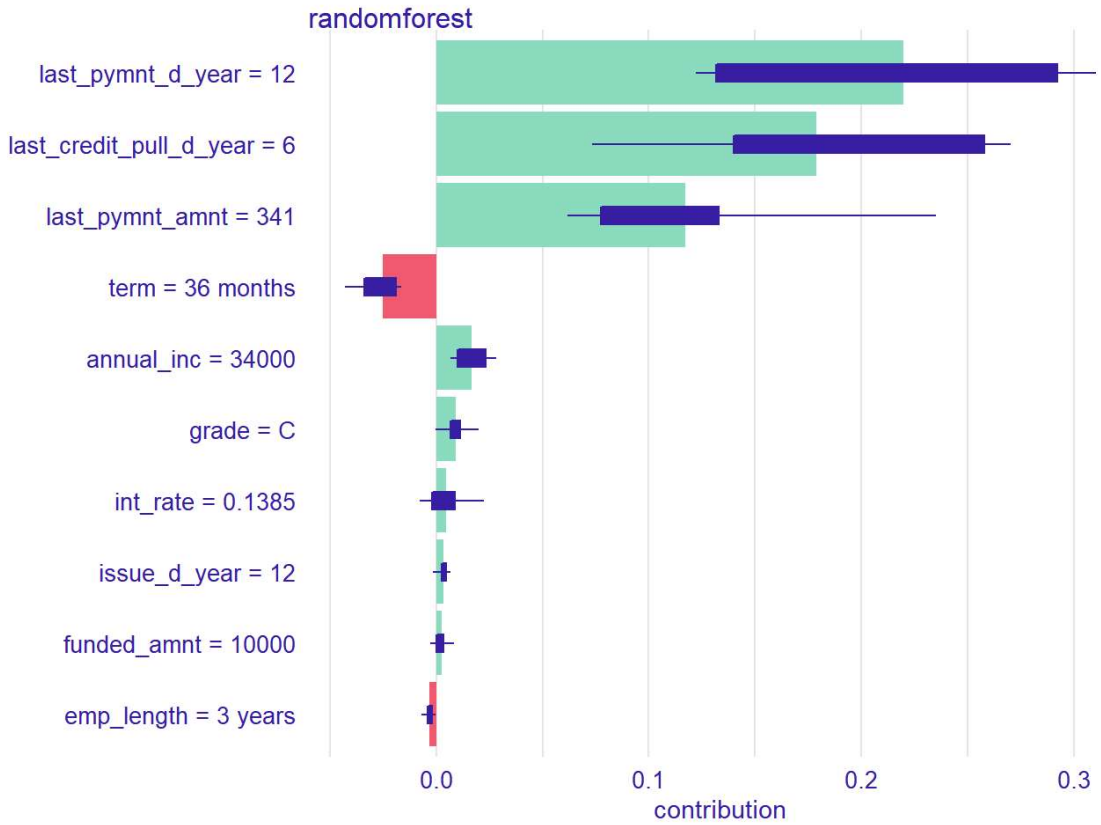


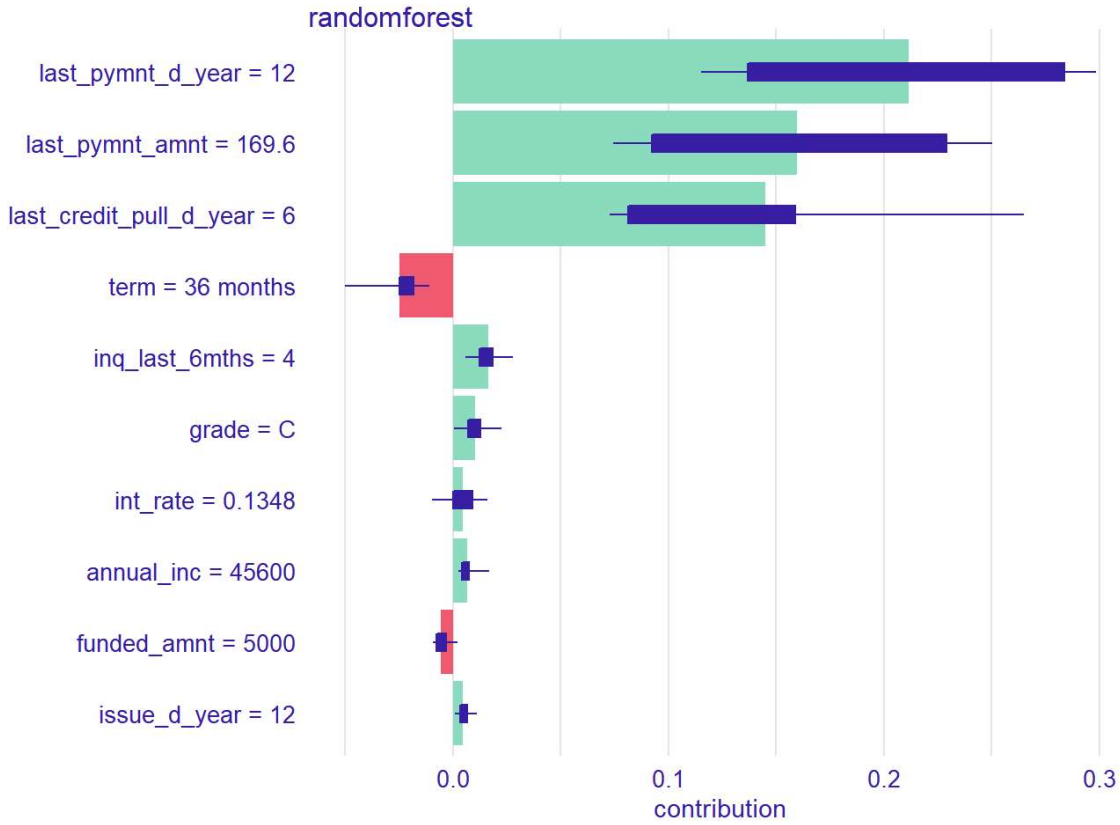

```
for (row in 1:nrow(top_5_fn)) {
    s_record <- top_5_tp[row,]
    explain_prediction(s_record)
}
```


randomforest
last_pymnt_amnt = 202.8
last_credit_pull_d_year = 6
term = 60 months
last_pymnt_d_year = 11
int_rate = 0.1793
inq_last_6mths = 5
grade = E
fico_range_low = 665
issue_d_year = 12
annual_inc = 50000
0.0
0.1
0.2
contribution


randomforest
last_pymnt_d_year = 12
last_pymnt_amnt = 337.7
last_credit_pull_d_year = 6
term = 36 months
issue_d_year = 13
grade = C
int_rate = 0.1316
fico_range_low = 685
inq_last_6mths = 0
annual_inc = 46000
0.0
0.1
0.2
0.3
contribution


randomforest
last_pymnt_d_year = 12
last_credit_pull_d_year = 6
last_pymnt_amnt = 337.4
term = 36 months
annual_inc = 40000
grade = C
int_rate = 0.1311
funded_amnt = 10000
fico_range_low = 705
loan_amnt = 10000
0.0
0.1
0.2
0.3
contribution


randomforest
last_pymnt_d_year = 12
last_credit_pull_d_year = 6
last_pymnt_amnt = 341
term = 36 months
annual_inc = 34000
grade = C
int_rate = 0.1385
issue_d_year = 12
funded_amnt = 10000
emp_length = 3 years
0.0
0.1
0.2
0.3
contribution



kaggle Prediction

```
kaggle_prediction <- predict(rf_wflow, loan_kaggle_decimal, type = "prob") %>%
  bind_cols(predict(rf_wflow, loan_kaggle_decimal, type = "class")) %>%
  bind_cols(loan_kaggle_decimal) %>%
  dplyr:::select.data.frame(id, loan_status = .pred_default)

head(kaggle_prediction)
```

id	loan_status
<dbl>	<dbl>
1077175	0.19430386
1075358	0.16555820
1075269	0.13288385
1071570	0.77480957
1064687	0.56165513
1065775	0.04974112

6 rows

```
kaggle_prediction %>% write_csv("Eagle Xuhui Ying_prediction.csv")
```