Project

Abstract

Lending Club was founded in 2006 as a peer-to-peer lending company that allows individual investor lenders to lend to other individual borrowers through an online platform. It services people that need personal loans between $1,000 and $40,000. The Lending Club data set represents thousands of loans made through the Lending Club platform.

The goal of this exploratory analysis is to apply most Machine Learning Algorithms we learn about in Stat 652 to the data from 2012-2014. Then compute the accuracy for each model to classify the Loan Status of the approved Lending Club loans, and select the best ML learning model for classifying *Loan Status*.

Introduction

Our target data is from year 2012 to 2014, which contains 423810 observations and 151 variables. 112 of them are numeric variables and 38 of them are character features. Among those character features, there are 34 categorical variables.

Rows	Columns	numeric	character	categorical
423810	151	112	38	34

numeric.vars	categorical.vars
loan_amnt	term
funded_amnt	grade
funded_amnt_inv	sub_grade
int_rate	emp_title
installment	emp_length
annual_inc	home_ownership
dti	verification_status
delinq_2yrs	issue_d
fico_range_low	loan_status
fico_range_high	pymnt_plan
inq_last_6mths	purpose
mths_since_last_delinq	title
mths_since_last_record	addr_state
open_acc	earliest_cr_line
pub_rec	initial_list_status
revol_bal	last_pymnt_d
revol_util	next_pymnt_d
total_acc	last_credit_pull_d

numeric.vars	categorical.vars
out_prncp	application_type
out_prncp_inv	verification_status_joint
total_pymnt	sec_app_earliest_cr_line
total_pymnt_inv	hardship_flag
total_rec_prncp	hardship_type
total_rec_int	hardship_reason
total_rec_late_fee	hardship_status
recoveries	hardship_start_date
collection_recovery_fee	hardship_end_date
last_pymnt_amnt	payment_plan_start_date
last_fico_range_high	hardship_loan_status
last_fico_range_low	disbursement_method
collections_12_mths_ex_med	debt_settlement_flag
mths_since_last_major_derog	debt_settlement_flag_date
policy_code	settlement_status
annual_inc_joint	settlement_date
dti_joint	
acc_now_delinq	
tot_coll_amt	
tot_cur_bal	
open_acc_6m	
open_act_il	
open_il_12m	
open_il_24m	
mths_since_rcnt_il	
total_bal_il	
il_util	
open_rv_12m	
open_rv_24m	
max_bal_bc	
all_util	
total_rev_hi_lim	
inq_fi	
total_cu_tl	
inq_last_12m	
acc_open_past_24mths	
avg_cur_bal	
bc_open_to_buy	
bc_util	
chargeoff_within_12_mths	
delinq_amnt	
mo_sin_old_il_acct	
mo_sin_old_rev_tl_op	
mo_sin_rcnt_rev_tl_op	
mo_sin_rcnt_tl	
mort_acc	
mths_since_recent_bc	
mths_since_recent_bc_dlq	
mths_since_recent_inq	
mths_since_recent_revol_delinq	
num_accts_ever_120_pd	
num_actv_bc_tl	

numeric.vars	categorical.vars
num_actv_rev_tl	
num_bc_sats	
num_bc_tl	
num_il_tl	
num_op_rev_tl	
num_rev_accts	
num_rev_tl_bal_gt_0	
num_sats	
num_tl_120dpd_2m	
num_tl_30dpd	
num_tl_90g_dpd_24m	
num_tl_op_past_12m	
pct_tl_nvr_dlq	
percent_bc_gt_75	
pub_rec_bankruptcies	
tax_liens	
tot_hi_cred_lim	
total_bal_ex_mort	
total_bc_limit	
total_il_high_credit_limit	
revol_bal_joint	
sec_app_fico_range_low	
sec_app_fico_range_high	
sec_app_inq_last_6mths	
sec_app_mort_acc	
sec_app_open_acc	
sec_app_revol_util	
sec_app_open_act_il	
sec_app_num_rev_accts	
sec_app_chargeoff_within_12_mths	
sec_app_collections_12_mths_ex_med	
sec_app_mths_since_last_major_derog	
deferral_term	
hardship_amount	
hardship_length	
hardship_dpd	
orig_projected_additional_accrued_interest	
hardship_payoff_balance_amount	
hardship_last_payment_amount	
settlement_amount	
settlement_percentage	
settlement_term	

Loan_status is our target response variables; it has 7 levels as the following shown:

loan_status	n	freq
Charged Off	70829	0.1671
Current	11925	0.0281
Default	1	0.0000
Fully Paid	340444	0.8033
In Grace Period	201	0.0005

loan_status	n	freq
Late (16-30 days)	73	0.0002
Late (31-120 days)	337	0.0008

It appears that 340,444 borrowers have fully paid the loan, which is 80% of the total borrowers.

To begin this exploratory analysis on the data and improve on the models' accuracy of predictions, we need to clean data first.

Data Cleaning and Wrangling

```
library(pacman)
p_load(tictoc, tidyverse, data.table, tidymodels, yardstick, janitor, naniar, discrim, gmodels, knitr)
```

We load the large data set and subset the data for the year from 2012 to 2014 at the beginning.

```
tic()
data <-
  fread("C:/Users/accepted_2007_to_2018Q4.csv",
              nThread = 12, na.strings = "")
toc()
```

```
## 24.86 sec elapsed
```

```
# subset the data for the years 2012-2014
club <- data %>%
  filter(str_detect(issue_d, '2012|2013|2014'))
```

We need to check if the data set contains any duplicate records and the situation of missing values before starting.

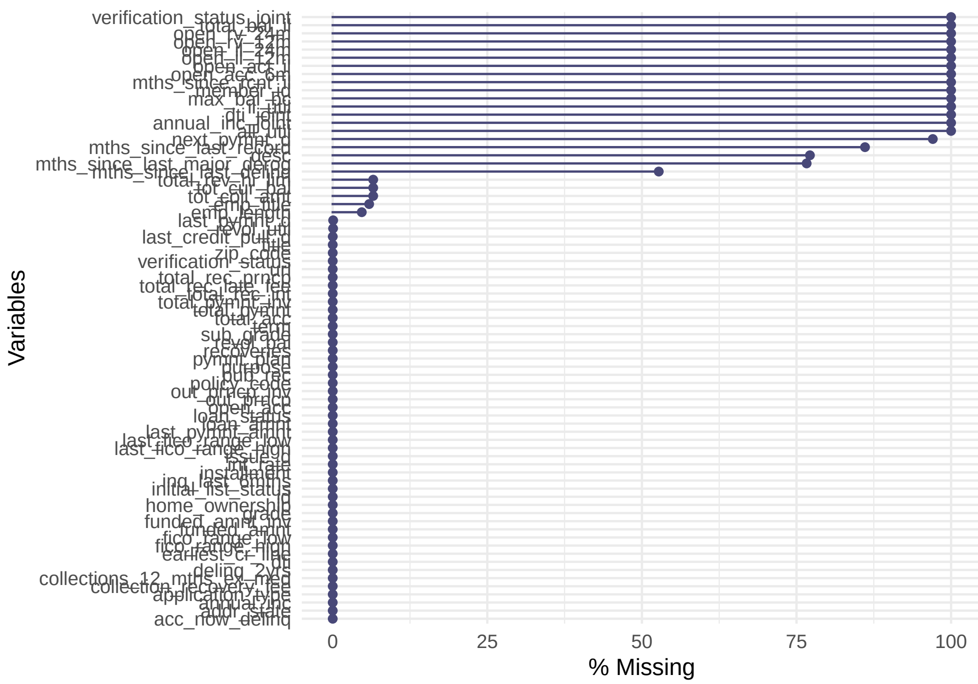
```
# check duplicate data
get_dupes(club)
```

```
## No variable names specified - using all columns.
```

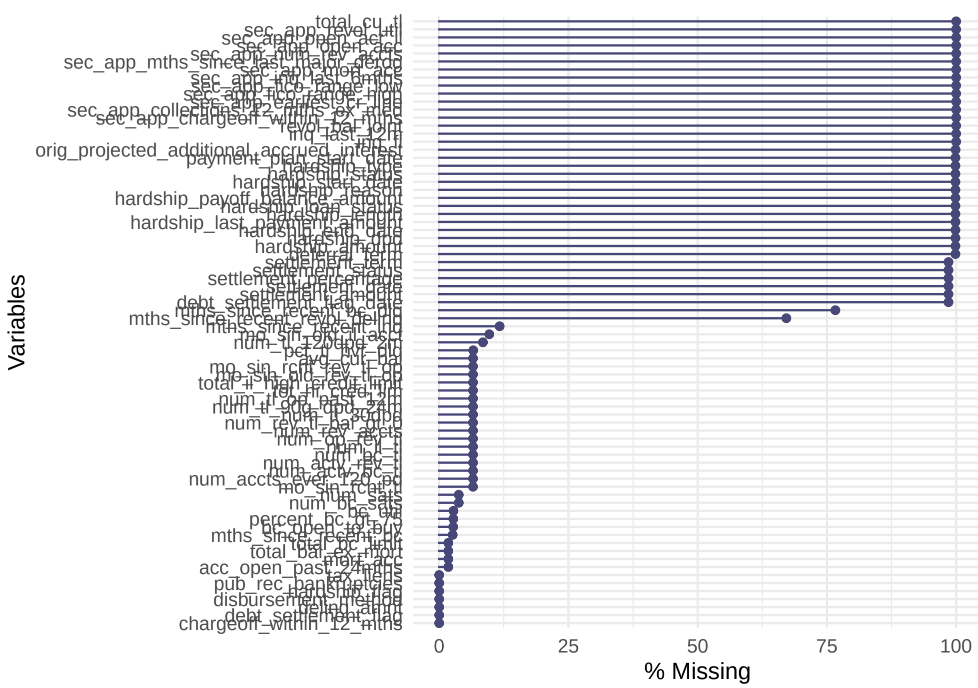
```
## No duplicate combinations found of: id, member_id, loan_amnt, funded_amnt, funded_amnt_inv, term, in
```

```
## Empty data.table (0 rows and 152 cols): id,member_id,loan_amnt,funded_amnt,funded_amnt_inv,term...
```

```
# check missing value
gg_miss_var(club[,1:75], show_pct = TRUE)
```




```r
gg_miss_var(club[,76:151], show_pct = TRUE)
```



Because there are many missing values, we plan to remove those variables whose proportion of missing values is greater than 0.5.

```r
# calculate the proportion of missing value for each variables and select which rate > 0.5
var_df <- club %>%
```

```r
  map_df(~ sum(is.na(.))/length(.)) %>%
  select_if(~ . > 0.5) %>%
  gather()

var_na <- club %>%
  map_df(~ sum(is.na(.))/length(.)) %>%
  select_if(~ . > 0.5) %>%
  names()
```

We build a classifier to *Loan_status*. We divided it to 2 levels for classification: *Fully Paid* and *Not Fully Paid*, and denote them as *"1"* and *"0"*, respectively.

```r
# select character variables and convert them to factor variables
#club %>%
  #select_if(is.character) %>%
  #map_df(~ as.factor(.))


club_df <- club %>%
  select(!one_of(var_na)) %>%                               # remove most missing value variables
  select(-id, -issue_d, -url, -zip_code, -policy_code, -application_type, -title, -emp_title,
         -earliest_cr_line, -last_credit_pull_d, -verification_status, -last_pymnt_d, -addr_state,
         -emp_length, -purpose) %>%
  mutate_if(is.character, as.factor) %>%                    # convert character vars to factor vars
  mutate(loan_status_level = ifelse(loan_status == "Fully Paid", 1,  0)) %>%      # classification
  mutate(loan_status_level = as.factor(loan_status_level))

str(club_df)
```

```
## Classes 'data.table' and 'data.frame':   423810 obs. of  79 variables:
##  $ loan_amnt                 : num  10400 15000 9600 7650 12800 ...
##  $ funded_amnt               : num  10400 15000 9600 7650 12800 ...
##  $ funded_amnt_inv           : num  10400 15000 9600 7650 12800 ...
##  $ term                      : Factor w/ 2 levels "36 months","60 months": 1 2 1 1 2 2 1 1 1 1 ...
##  $ int_rate                  : num  6.99 12.39 13.66 13.66 17.14 ...
##  $ installment               : num  321 337 327 260 319 ...
##  $ grade                     : Factor w/ 7 levels "A","B","C","D",..: 1 3 3 3 4 4 3 3 2 4 ...
##  $ sub_grade                 : Factor w/ 35 levels "A1","A2","A3",..: 3 11 13 13 19 16 13 14 10 20 .
##  $ home_ownership            : Factor w/ 6 levels "ANY","MORTGAGE",..: 2 6 6 6 2 6 2 6 2 6 ...
##  $ annual_inc                : num  58000 78000 69000 50000 125000 63800 75000 72000 89000 60000 ...
##  $ loan_status               : Factor w/ 7 levels "Charged Off",..: 1 4 4 1 2 4 4 1 4 1 ...
##  $ pymnt_plan                : Factor w/ 2 levels "n","y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ dti                       : num  14.92 12.03 25.81 34.81 8.31 ...
##  $ delinq_2yrs               : num  0 0 0 0 1 0 0 1 0 0 ...
##  $ fico_range_low            : num  710 750 680 685 665 685 675 665 685 680 ...
##  $ fico_range_high           : num  714 754 684 689 669 689 679 669 689 684 ...
##  $ inq_last_6mths            : num  2 0 0 1 0 0 0 0 1 0 ...
##  $ open_acc                  : num  17 6 12 11 8 10 7 14 9 11 ...
##  $ pub_rec                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ revol_bal                 : num  6133 138008 16388 16822 5753 ...
##  $ revol_util                : num  31.6 29 59.4 91.9 100.9 ...
##  $ total_acc                 : num  36 17 44 20 13 35 31 23 32 19 ...
##  $ initial_list_status       : Factor w/ 2 levels "f","w": 2 2 1 1 2 2 1 1 1 1 ...
```

```
##  $ out_prncp                 : num  0 0 0 0 2969 ...
##  $ out_prncp_inv             : num  0 0 0 0 2969 ...
##  $ total_pymnt               : num  6612 17392 9973 2282 15994 ...
##  $ total_pymnt_inv           : num  6612 17392 9973 2282 15994 ...
##  $ total_rec_prncp           : num  5218 15000 9600 704 9831 ...
##  $ total_rec_int             : num  873 2392 373 340 6163 ...
##  $ total_rec_late_fee        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ recoveries                : num  521 0 0 1238 0 ...
##  $ collection_recovery_fee   : num  93.8 0 0 222.8 0 ...
##  $ last_pymnt_amnt           : num  321.1 12017.8 9338.6 17.7 319.1 ...
##  $ last_fico_range_high      : num  564 704 679 559 664 529 719 499 789 664 ...
##  $ last_fico_range_low       : num  560 700 675 555 660 525 715 0 785 660 ...
##  $ collections_12_mths_ex_med: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ acc_now_delinq            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ tot_coll_amt              : num  0 0 0 0 0 0 0 0 0 900 ...
##  $ tot_cur_bal               : num  162110 149140 38566 64426 261815 ...
##  $ total_rev_hi_lim          : num  19400 184500 27600 18300 5700 ...
##  $ acc_open_past_24mths      : num  7 5 8 6 2 4 2 6 6 7 ...
##  $ avg_cur_bal               : num  9536 29828 3214 5857 32727 ...
##  $ bc_open_to_buy            : num  7599 9525 6494 332 0 ...
##  $ bc_util                   : num  41.5 4.7 69.2 93.2 103.2 ...
##  $ chargeoff_within_12_mths  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ delinq_amnt               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ mo_sin_old_il_acct        : num  76 103 183 137 16 135 93 132 158 191 ...
##  $ mo_sin_old_rev_tl_op      : num  290 244 265 148 170 136 167 194 148 122 ...
##  $ mo_sin_rcnt_rev_tl_op     : num  1 1 23 8 21 7 21 15 24 2 ...
##  $ mo_sin_rcnt_tl            : num  1 1 3 8 16 7 10 12 6 2 ...
##  $ mort_acc                  : num  1 0 0 0 5 0 2 6 5 0 ...
##  $ mths_since_recent_bc      : num  5 47 24 17 21 7 27 23 24 6 ...
##  $ mths_since_recent_inq     : num  1 NA 17 3 1 7 NA 16 2 1 ...
##  $ num_accts_ever_120_pd     : num  4 0 0 0 1 1 2 0 0 0 ...
##  $ num_actv_bc_tl            : num  6 1 4 1 3 3 1 3 3 3 ...
##  $ num_actv_rev_tl           : num  9 4 7 4 5 4 3 5 4 8 ...
##  $ num_bc_sats               : num  7 1 5 1 3 3 1 7 3 3 ...
##  $ num_bc_tl                 : num  18 2 16 4 5 12 7 9 6 6 ...
##  $ num_il_tl                 : num  2 8 17 12 1 16 4 4 17 6 ...
##  $ num_op_rev_tl             : num  14 5 8 4 5 5 4 10 4 9 ...
##  $ num_rev_accts             : num  32 9 26 8 7 18 25 13 10 13 ...
##  $ num_rev_tl_bal_gt_0       : num  9 4 7 4 5 4 3 5 4 8 ...
##  $ num_sats                  : num  17 6 12 11 8 10 7 14 9 11 ...
##  $ num_tl_120dpd_2m          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num_tl_30dpd              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num_tl_90g_dpd_24m        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num_tl_op_past_12m        : num  4 4 3 2 0 2 1 1 1 4 ...
##  $ pct_tl_nvr_dlq            : num  83.3 100 100 100 76.9 91.4 87.1 95.7 96.8 89.5 ...
##  $ percent_bc_gt_75          : num  14.3 0 60 100 100 100 100 66.7 66.7 0 ...
##  $ pub_rec_bankruptcies      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ tax_liens                 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ tot_hi_cred_lim           : num  179407 196500 52490 82331 368700 ...
##  $ total_bal_ex_mort         : num  15030 149140 38566 64426 18007 ...
##  $ total_bc_limit            : num  13000 10000 21100 4900 4400 15000 4000 36000 9800 5500 ...
##  $ total_il_high_credit_limit: num  11325 12000 24890 64031 18000 ...
##  $ hardship_flag             : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ disbursement_method       : Factor w/ 1 level "Cash": 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ debt_settlement_flag      : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ loan_status_level         : Factor w/ 2 levels "0","1": 1 2 2 1 1 2 2 1 2 1 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Here is the summary of classification for the variable *loan_status*:

```
# check loan status level
club_df %>%
  group_by(loan_status, loan_status_level) %>%
  tally()
```

```
## # A tibble: 7 x 3
## # Groups:   loan_status [7]
##   loan_status        loan_status_level      n
##   <fct>              <fct>              <int>
## 1 Charged Off        0                  70829
## 2 Current            0                  11925
## 3 Default            0                      1
## 4 Fully Paid         1                 340444
## 5 In Grace Period    0                    201
## 6 Late (16-30 days)  0                     73
## 7 Late (31-120 days) 0                    337
```

```
CrossTable(club_df$loan_status_level, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  423810
##
##
##           |         0 |         1 |
##           |-----------|-----------|
##           |     83366 |    340444 |
##           |     0.197 |     0.803 |
##           |-----------|-----------|
##
##
##
##
```

Because the data set for year 2012-2014 is pretty huge, we take some sample that is 10% of data set for modeling.

```r
# sample 10% of data set
set.seed(999)
club_samp <- club_df %>%
  slice_sample(n = 0.1*nrow(club))
```

And the following is the summary of classification for the variable *loan_status* in our overall data set.

```r
#summarize the y-variable
CrossTable(club_samp$loan_status_level, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  42381
##
##
##           |         0 |         1 |
##           |-----------|-----------|
##           |      8360 |     34021 |
##           |     0.197 |     0.803 |
##           |-----------|-----------|
##
##
##
##
```

We use a 75-25 split to create Training and Testing data sets.

```r
# split the training and testing dataset
set.seed(999)
samp_split <- club_samp %>%
  initial_split(prop = 0.75)

samp_split
```

```
## <Analysis/Assess/Total>
## <31786/10595/42381>
```

We use *recipe* function to clean up and process our final data.

```r
samp_recipe <- training(samp_split) %>%
  recipe(loan_status_level ~ .) %>%
  step_rm(loan_status) %>%
  step_nzv(all_predictors()) %>%
  step_knnimpute(all_predictors()) %>%
```

```r
  prep()

summary(samp_recipe)
```

```
## # A tibble: 61 x 4
##    variable        type    role      source
##    <chr>           <chr>   <chr>     <chr>
##  1 loan_amnt       numeric predictor original
##  2 funded_amnt     numeric predictor original
##  3 funded_amnt_inv numeric predictor original
##  4 term            nominal predictor original
##  5 int_rate        numeric predictor original
##  6 installment     numeric predictor original
##  7 grade           nominal predictor original
##  8 sub_grade       nominal predictor original
##  9 home_ownership  nominal predictor original
## 10 annual_inc      numeric predictor original
## # ... with 51 more rows
```

```r
tidy(samp_recipe)
```

```
## # A tibble: 3 x 6
##   number operation type      trained skip  id
##    <int> <chr>     <chr>     <lgl>   <lgl> <chr>
## 1      1 step      rm        TRUE    FALSE rm_XPPN6
## 2      2 step      nzv       TRUE    FALSE nzv_0Xq94
## 3      3 step      knnimpute TRUE    FALSE knnimpute_5S9Hw
```

```r
samp_testing <- samp_recipe %>%
  bake(testing(samp_split))

samp_testing
```

```
## # A tibble: 10,595 x 61
##    loan_amnt funded_amnt funded_amnt_inv term      int_rate installment grade
##        <dbl>       <dbl>           <dbl> <fct>        <dbl>       <dbl> <fct>
##  1      9100        9100            9100 36 months    18.8         333. D
##  2      6000        6000            6000 36 months    14.1         205. B
##  3     11000       11000           11000 60 months    19.0         285. E
##  4     16000       16000           16000 36 months    11.0         524. B
##  5     20000       20000           20000 60 months    16.8         495. C
##  6      8000        8000            8000 36 months    14.5         275. C
##  7      9500        9500            9500 36 months    10.6         309. B
##  8     18000       18000           18000 60 months    16.2         440. C
##  9      5000        5000            5000 36 months     7.12        155. A
## 10     12000       12000           12000 36 months    10.2         388. B
## # ... with 10,585 more rows, and 54 more variables: sub_grade <fct>,
## #   home_ownership <fct>, annual_inc <dbl>, dti <dbl>, delinq_2yrs <dbl>,
## #   fico_range_low <dbl>, fico_range_high <dbl>, inq_last_6mths <dbl>,
## #   open_acc <dbl>, pub_rec <dbl>, revol_bal <dbl>, revol_util <dbl>,
## #   total_acc <dbl>, initial_list_status <fct>, total_pymnt <dbl>,
```

```
## #   total_pymnt_inv <dbl>, total_rec_prncp <dbl>, total_rec_int <dbl>,
## #   recoveries <dbl>, collection_recovery_fee <dbl>, last_pymnt_amnt <dbl>,
## #   last_fico_range_high <dbl>, last_fico_range_low <dbl>, tot_cur_bal <dbl>,
## #   total_rev_hi_lim <dbl>, acc_open_past_24mths <dbl>, avg_cur_bal <dbl>,
## #   bc_open_to_buy <dbl>, bc_util <dbl>, mo_sin_old_il_acct <dbl>,
## #   mo_sin_old_rev_tl_op <dbl>, mo_sin_rcnt_rev_tl_op <dbl>,
## #   mo_sin_rcnt_tl <dbl>, mort_acc <dbl>, mths_since_recent_bc <dbl>,
## #   mths_since_recent_inq <dbl>, num_accts_ever_120_pd <dbl>,
## #   num_actv_bc_tl <dbl>, num_actv_rev_tl <dbl>, num_bc_sats <dbl>,
## #   num_bc_tl <dbl>, num_il_tl <dbl>, num_op_rev_tl <dbl>, num_rev_accts <dbl>,
## #   num_rev_tl_bal_gt_0 <dbl>, num_sats <dbl>, num_tl_op_past_12m <dbl>,
## #   percent_bc_gt_75 <dbl>, pub_rec_bankruptcies <dbl>, tot_hi_cred_lim <dbl>,
## #   total_bal_ex_mort <dbl>, total_bc_limit <dbl>,
## #   total_il_high_credit_limit <dbl>, loan_status_level <fct>
```

```
samp_training <- juice(samp_recipe)

samp_training
```

```
## # A tibble: 31,786 x 61
##    loan_amnt funded_amnt funded_amnt_inv term      int_rate installment grade
##        <dbl>       <dbl>           <dbl> <fct>        <dbl>       <dbl> <fct>
##  1      5000        5000            5000 36 months    12.0         166. B
##  2     20000       20000           20000 36 months    21           754. E
##  3      6400        6400            6400 36 months     6.99        198. A
##  4     24250       24250           24250 60 months    11.4         533. B
##  5     23975       23975           23975 60 months    22.2         664. E
##  6      8000        8000            8000 36 months    12.5         268. B
##  7     15950       15950           15950 36 months    17.0         569. D
##  8     15000       15000           14950 60 months    16.2         366. C
##  9     15500       15500           15500 36 months    11.0         507. B
## 10     10000       10000           10000 60 months    17.0         248. D
## # ... with 31,776 more rows, and 54 more variables: sub_grade <fct>,
## #   home_ownership <fct>, annual_inc <dbl>, dti <dbl>, delinq_2yrs <dbl>,
## #   fico_range_low <dbl>, fico_range_high <dbl>, inq_last_6mths <dbl>,
## #   open_acc <dbl>, pub_rec <dbl>, revol_bal <dbl>, revol_util <dbl>,
## #   total_acc <dbl>, initial_list_status <fct>, total_pymnt <dbl>,
## #   total_pymnt_inv <dbl>, total_rec_prncp <dbl>, total_rec_int <dbl>,
## #   recoveries <dbl>, collection_recovery_fee <dbl>, last_pymnt_amnt <dbl>,
## #   last_fico_range_high <dbl>, last_fico_range_low <dbl>, tot_cur_bal <dbl>,
## #   total_rev_hi_lim <dbl>, acc_open_past_24mths <dbl>, avg_cur_bal <dbl>,
## #   bc_open_to_buy <dbl>, bc_util <dbl>, mo_sin_old_il_acct <dbl>,
## #   mo_sin_old_rev_tl_op <dbl>, mo_sin_rcnt_rev_tl_op <dbl>,
## #   mo_sin_rcnt_tl <dbl>, mort_acc <dbl>, mths_since_recent_bc <dbl>,
## #   mths_since_recent_inq <dbl>, num_accts_ever_120_pd <dbl>,
## #   num_actv_bc_tl <dbl>, num_actv_rev_tl <dbl>, num_bc_sats <dbl>,
## #   num_bc_tl <dbl>, num_il_tl <dbl>, num_op_rev_tl <dbl>, num_rev_accts <dbl>,
## #   num_rev_tl_bal_gt_0 <dbl>, num_sats <dbl>, num_tl_op_past_12m <dbl>,
## #   percent_bc_gt_75 <dbl>, pub_rec_bankruptcies <dbl>, tot_hi_cred_lim <dbl>,
## #   total_bal_ex_mort <dbl>, total_bc_limit <dbl>,
## #   total_il_high_credit_limit <dbl>, loan_status_level <fct>
```

After finishing all these steps, we start to training some models on the data.

Model 0: Null Model

Setup the Model

```
mod_null <- null_model () %>%
  set_engine("parsnip") %>%
  set_mode("classification") %>%
  fit(loan_status_level ~ ., data = samp_training)

mod_null
```

```
## parsnip model object
##
## Fit time:  0ms
## Null Regression Model
## Predicted Value: 1
```

Evaluating Model Performance

```
# prediction on testing data set
pred_null <- mod_null %>%
  predict(samp_testing) %>%
  bind_cols(samp_testing)

pred_null %>%
  conf_mat(truth = loan_status_level, estimate = .pred_class)
```

```
##           Truth
## Prediction    0    1
##          0    0    0
##          1 2148 8447
```

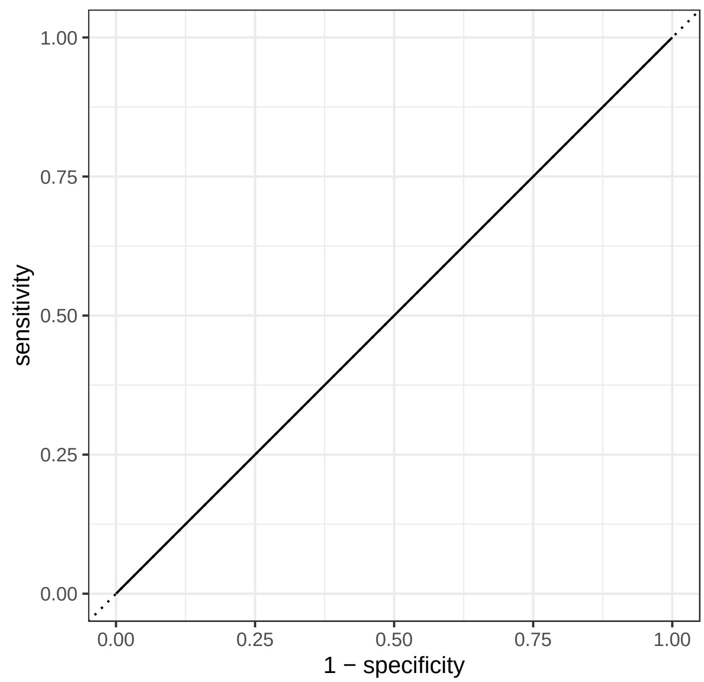
```
pred_null %>%
  metrics(truth = loan_status_level, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.797
## 2 kap      binary         0
```

```
# plot
plot_null <- mod_null %>%
  predict(samp_testing, type = "prob") %>%
  bind_cols(samp_testing)

plot_null %>%
  roc_auc(loan_status_level, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary           0.5
```

```
plot_null %>%
  roc_curve(loan_status_level, .pred_1) %>%
  autoplot()
```



Model 1: kNN

Data Clean up for kNN

```
# filter numeric variables for kNN
var_num <- club_samp %>%
  select_if(is.numeric) %>%
  names()

samp_recipe_knn <- training(samp_split) %>%
  select(loan_status_level, one_of(var_num)) %>%
  recipe(loan_status_level ~ .) %>%
  step_nzv(all_predictors()) %>%
  step_knnimpute(all_predictors()) %>%
  step_normalize(all_predictors()) %>%
  prep()

summary(samp_recipe_knn)
```

```
## # A tibble: 56 x 4
##    variable        type    role      source
##    <chr>           <chr>   <chr>     <chr>
##  1 loan_amnt       numeric predictor original
##  2 funded_amnt     numeric predictor original
##  3 funded_amnt_inv numeric predictor original
##  4 int_rate        numeric predictor original
##  5 installment     numeric predictor original
##  6 annual_inc      numeric predictor original
##  7 dti             numeric predictor original
##  8 delinq_2yrs     numeric predictor original
##  9 fico_range_low  numeric predictor original
## 10 fico_range_high numeric predictor original
## # ... with 46 more rows
```

```
tidy(samp_recipe_knn)
```

```
## # A tibble: 3 x 6
##   number operation type      trained skip  id
##    <int> <chr>     <chr>     <lgl>   <lgl> <chr>
## 1      1 step      nzv       TRUE    FALSE nzv_AyHtf
## 2      2 step      knnimpute TRUE    FALSE knnimpute_Wrhhx
## 3      3 step      normalize TRUE    FALSE normalize_Mzwjr
```

```
samp_testing_knn <- samp_recipe_knn %>%
  bake(testing(samp_split))

samp_testing_knn
```

```
## # A tibble: 10,595 x 56
##    loan_amnt funded_amnt funded_amnt_inv int_rate installment annual_inc    dti
##        <dbl>       <dbl>           <dbl>    <dbl>       <dbl>      <dbl>  <dbl>
##  1    -0.673      -0.673          -0.673   1.11       -0.456      -0.865 -0.222
##  2    -1.05       -1.05           -1.05    0.0185     -0.985      -0.266 -0.407
##  3    -0.443      -0.443          -0.442   1.14       -0.654      -0.611 -0.903
##  4     0.164       0.164           0.165  -0.690       0.334       0.115 -0.523
##  5     0.649       0.649           0.650   0.634       0.214       0.823 -0.683
##  6    -0.807      -0.807          -0.806   0.110      -0.695      -0.357  1.08
##  7    -0.625      -0.625          -0.624  -0.770      -0.554      -0.157 -1.39
##  8     0.406       0.406           0.407   0.501      -0.0143     -0.103  0.773
##  9    -1.17       -1.17           -1.17   -1.58       -1.19       -0.937  0.689
## 10    -0.322      -0.322          -0.321  -0.882      -0.228      -0.574 -0.240
## # ... with 10,585 more rows, and 49 more variables: delinq_2yrs <dbl>,
## #   fico_range_low <dbl>, fico_range_high <dbl>, inq_last_6mths <dbl>,
## #   open_acc <dbl>, pub_rec <dbl>, revol_bal <dbl>, revol_util <dbl>,
## #   total_acc <dbl>, total_pymnt <dbl>, total_pymnt_inv <dbl>,
## #   total_rec_prncp <dbl>, total_rec_int <dbl>, recoveries <dbl>,
## #   collection_recovery_fee <dbl>, last_pymnt_amnt <dbl>,
## #   last_fico_range_high <dbl>, last_fico_range_low <dbl>, tot_cur_bal <dbl>,
## #   total_rev_hi_lim <dbl>, acc_open_past_24mths <dbl>, avg_cur_bal <dbl>,
## #   bc_open_to_buy <dbl>, bc_util <dbl>, mo_sin_old_il_acct <dbl>,
## #   mo_sin_old_rev_tl_op <dbl>, mo_sin_rcnt_rev_tl_op <dbl>,
## #   mo_sin_rcnt_tl <dbl>, mort_acc <dbl>, mths_since_recent_bc <dbl>,
```

```
## #   mths_since_recent_inq <dbl>, num_accts_ever_120_pd <dbl>,
## #   num_actv_bc_tl <dbl>, num_actv_rev_tl <dbl>, num_bc_sats <dbl>,
## #   num_bc_tl <dbl>, num_il_tl <dbl>, num_op_rev_tl <dbl>, num_rev_accts <dbl>,
## #   num_rev_tl_bal_gt_0 <dbl>, num_sats <dbl>, num_tl_op_past_12m <dbl>,
## #   percent_bc_gt_75 <dbl>, pub_rec_bankruptcies <dbl>, tot_hi_cred_lim <dbl>,
## #   total_bal_ex_mort <dbl>, total_bc_limit <dbl>,
## #   total_il_high_credit_limit <dbl>, loan_status_level <fct>
```

```r
samp_training_knn <- juice(samp_recipe_knn)

samp_training_knn
```

```
## # A tibble: 31,786 x 56
##    loan_amnt funded_amnt funded_amnt_inv int_rate installment annual_inc
##        <dbl>       <dbl>           <dbl>    <dbl>       <dbl>      <dbl>
##  1   -1.17       -1.17           -1.17     -0.462      -1.15     -0.683
##  2    0.649       0.649           0.650     1.60        1.29      0.351
##  3   -1.00       -1.00           -1.00     -1.60       -1.02      1.39
##  4    1.16        1.16            1.17     -0.587       0.371     0.660
##  5    1.13        1.13            1.13      1.86        0.916    -0.0300
##  6   -0.807      -0.807          -0.806    -0.347      -0.727    -0.629
##  7    0.158       0.158           0.159     0.682       0.520    -0.611
##  8    0.0424      0.0424          0.0373    0.501      -0.318     0.0970
##  9    0.103       0.103           0.104    -0.690       0.266    -0.647
## 10   -0.564      -0.564          -0.564     0.682      -0.806    -0.520
## # ... with 31,776 more rows, and 50 more variables: dti <dbl>,
## #   delinq_2yrs <dbl>, fico_range_low <dbl>, fico_range_high <dbl>,
## #   inq_last_6mths <dbl>, open_acc <dbl>, pub_rec <dbl>, revol_bal <dbl>,
## #   revol_util <dbl>, total_acc <dbl>, total_pymnt <dbl>,
## #   total_pymnt_inv <dbl>, total_rec_prncp <dbl>, total_rec_int <dbl>,
## #   recoveries <dbl>, collection_recovery_fee <dbl>, last_pymnt_amnt <dbl>,
## #   last_fico_range_high <dbl>, last_fico_range_low <dbl>, tot_cur_bal <dbl>,
## #   total_rev_hi_lim <dbl>, acc_open_past_24mths <dbl>, avg_cur_bal <dbl>,
## #   bc_open_to_buy <dbl>, bc_util <dbl>, mo_sin_old_il_acct <dbl>,
## #   mo_sin_old_rev_tl_op <dbl>, mo_sin_rcnt_rev_tl_op <dbl>,
## #   mo_sin_rcnt_tl <dbl>, mort_acc <dbl>, mths_since_recent_bc <dbl>,
## #   mths_since_recent_inq <dbl>, num_accts_ever_120_pd <dbl>,
## #   num_actv_bc_tl <dbl>, num_actv_rev_tl <dbl>, num_bc_sats <dbl>,
## #   num_bc_tl <dbl>, num_il_tl <dbl>, num_op_rev_tl <dbl>, num_rev_accts <dbl>,
## #   num_rev_tl_bal_gt_0 <dbl>, num_sats <dbl>, num_tl_op_past_12m <dbl>,
## #   percent_bc_gt_75 <dbl>, pub_rec_bankruptcies <dbl>, tot_hi_cred_lim <dbl>,
## #   total_bal_ex_mort <dbl>, total_bc_limit <dbl>,
## #   total_il_high_credit_limit <dbl>, loan_status_level <fct>
```

Setup the Model

```r
mod_knn <- nearest_neighbor(neighbors = 13) %>%
  set_engine("kknn") %>%
  set_mode("classification") %>%
  fit(loan_status_level ~ ., data = samp_training_knn)

mod_knn
```

```
## parsnip model object
##
## Fit time:  1m 21.4s
##
## Call:
## kknn::train.kknn(formula = loan_status_level ~ ., data = data,     ks = min_rows(13, data, 5))
##
## Type of response variable: nominal
## Minimal misclassification: 0.09884855
## Best kernel: optimal
## Best k: 13
```

Evaluating Model Performance

```r
# prediction on testing data set
pred_knn <- mod_knn %>%
  predict(samp_testing_knn) %>%
  bind_cols(samp_testing_knn)

pred_knn %>%
  conf_mat(truth = loan_status_level, estimate = .pred_class)
```

```
##           Truth
## Prediction    0    1
##          0 1202  103
##          1  946 8344
```

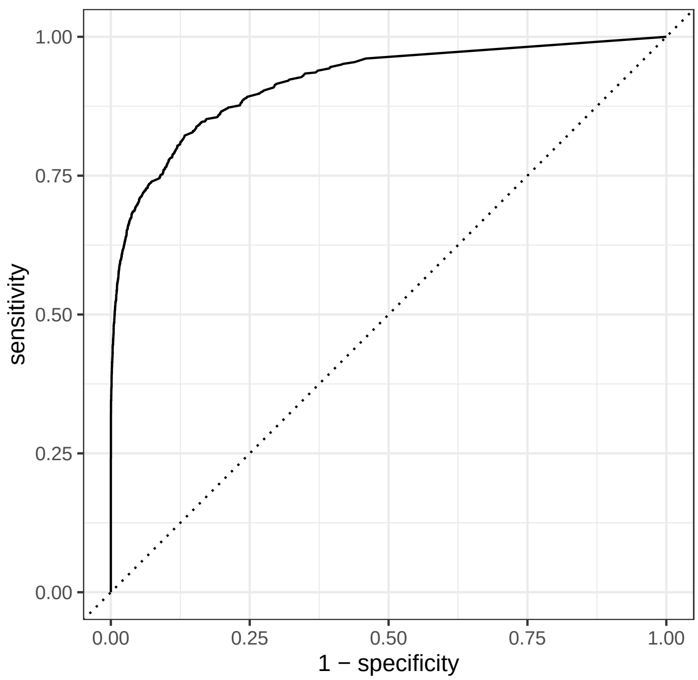
```r
pred_knn %>%
  metrics(truth = loan_status_level, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.901
## 2 kap      binary         0.641
```

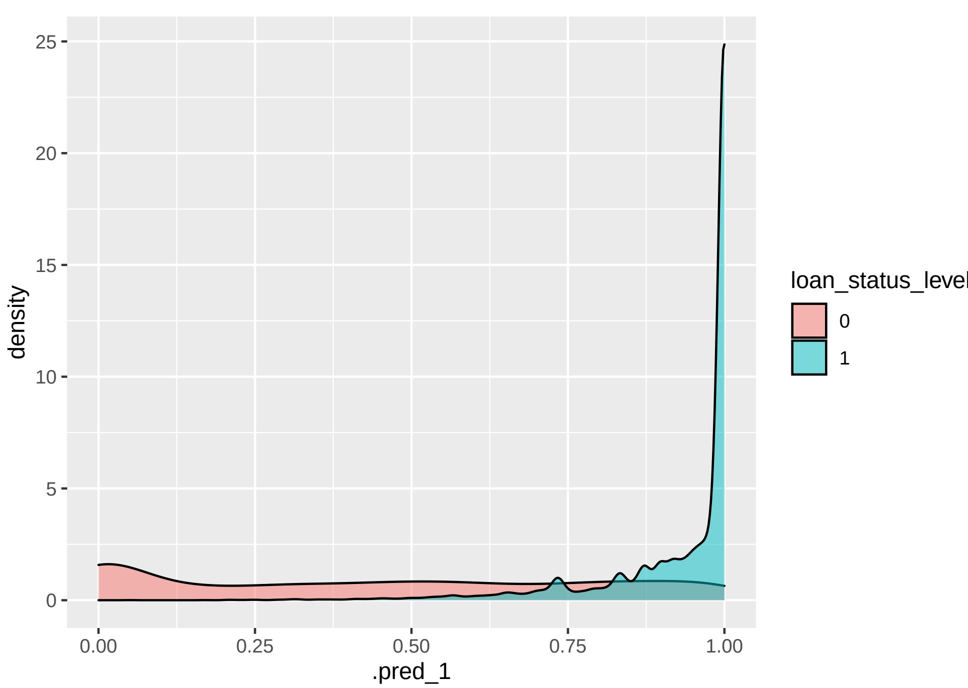
```r
# plot
plot_knn <- mod_knn %>%
  predict(samp_testing_knn, type = "prob") %>%
  bind_cols(samp_testing_knn)

plot_knn %>%
  roc_auc(loan_status_level, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.919
```

```
plot_knn %>%
  roc_curve(loan_status_level, .pred_0) %>%
  autoplot()
```




```
plot_knn %>%
  ggplot() + geom_density(aes(x = .pred_1, fill = loan_status_level), alpha = 0.5)
```



Improving Model Performance

```
# knn tuning to chose the best K
knn_model <- nearest_neighbor(neighbors = tune()) %>%
  set_engine("kknn") %>%
  set_mode("classification")

knn_wflow <- workflow() %>%
  add_recipe(samp_recipe_knn) %>%
  add_model(knn_model)

folds <- vfold_cv(training(samp_split), v = 10)

knn_grid <- seq(5, 50, by = 2)

knn_tune_resultes <- knn_wflow %>%
  tune_grid(resamples = folds, grid = knn_grid)

#knn_tune_resultes %>%
  #collect_metrics()

knn_trees <- knn_tune_resultes %>%
  select_best("accuracy")

knn_trees
```

```
## # A tibble: 1 x 2
##   neighbors .config
##       <int> <chr>
## 1        12 Preprocessor1_Model5
```

```
knn_acc <- knn_wflow %>%
  finalize_workflow(knn_trees) %>%
  last_fit(samp_split) %>%
  collect_metrics()

knn_acc
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy binary         0.901 Preprocessor1_Model1
## 2 roc_auc  binary         0.916 Preprocessor1_Model1
```

Model 2: Boosted C5.0

Setup the Model

```
mod_C50 <- boost_tree(trees = 100) %>%
  set_engine("C5.0") %>%
```

```r
  set_mode("classification") %>%
  fit(loan_status_level ~ ., data = samp_training)

mod_C50
```

```
## parsnip model object
##
## Fit time:  1m 29.5s
##
## Call:
## C5.0.default(x = x, y = y, trials = 100, control = C50::C5.0Control(minCases
##  = 2, sample = 0))
##
## Classification Tree
## Number of samples: 31786
## Number of predictors: 60
##
## Number of boosting iterations: 100
## Average tree size: 96.8
##
## Non-standard options: attempt to group attributes
```

Evaluating Model Performance

```r
# prediction on testing data set
pred_C50 <- mod_C50 %>%
  predict(samp_testing) %>%
  bind_cols(samp_testing)

pred_C50 %>%
  conf_mat(truth = loan_status_level, estimate = .pred_class)
```

```
##           Truth
## Prediction    0    1
##          0 2089   11
##          1   59 8436
```

```r
pred_C50 %>%
  metrics(truth = loan_status_level, estimate = .pred_class)
```

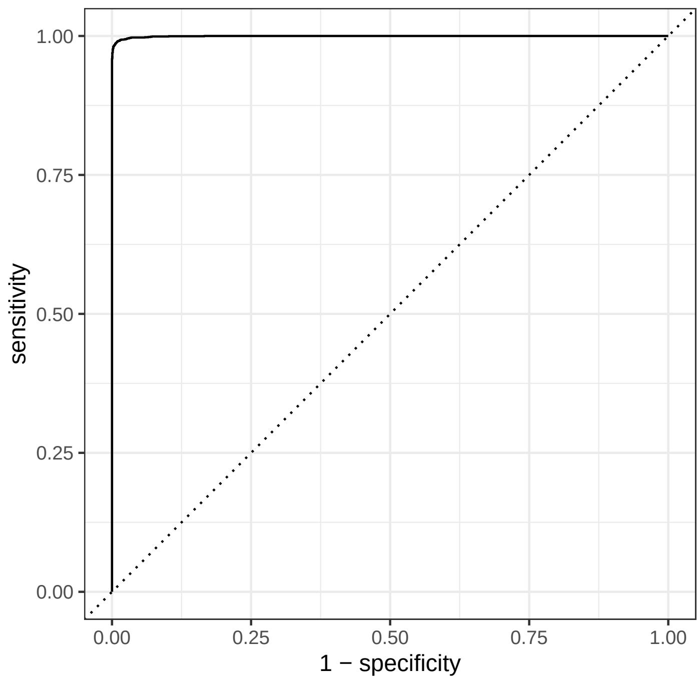
```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.993
## 2 kap      binary         0.979
```

```r
# plot
plot_C50 <- mod_C50 %>%
  predict(samp_testing, type = "prob") %>%
  bind_cols(samp_testing)
```

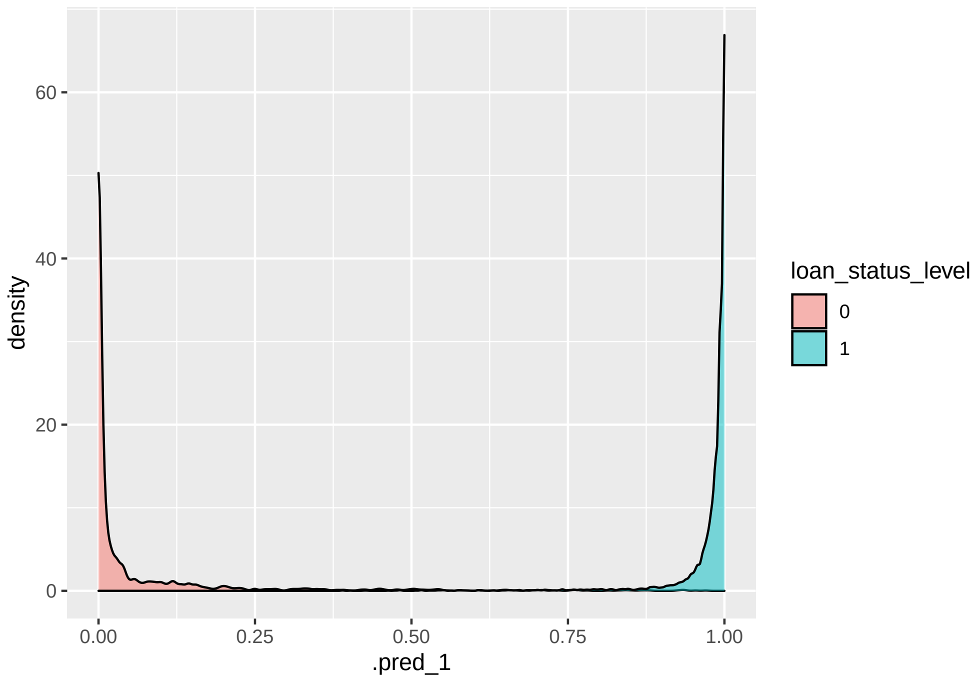
```
plot_C50 %>%
  roc_auc(loan_status_level, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary          1.00
```

```
plot_C50 %>%
  roc_curve(loan_status_level, .pred_0) %>%
  autoplot()
```




```
plot_C50 %>%
  ggplot() + geom_density(aes(x = .pred_1, fill = loan_status_level), alpha = 0.5)
```



Model 3: Random Forest

Setup the Model

```
mod_ranger <- rand_forest(trees = 300) %>%
  set_engine("ranger") %>%
  set_mode("classification") %>%
  fit(loan_status_level ~ ., data = samp_training)

mod_ranger
```

```
## parsnip model object
##
## Fit time:  29.8s
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, num.trees = ~300,      num.threads = 1, verbose = FA
##
## Type:                             Probability estimation
## Number of trees:                  300
## Sample size:                      31786
## Number of independent variables:  60
## Mtry:                             7
## Target node size:                 10
## Variable importance mode:         none
## Splitrule:                        gini
## OOB prediction error (Brier s.):  0.01207474
```

Evaluating Model Performance

```r
# prediction on testing data set
pred_ranger <- mod_ranger %>%
  predict(samp_testing) %>%
  bind_cols(samp_testing)

pred_ranger %>%
  conf_mat(truth = loan_status_level, estimate = .pred_class)
```

```
##           Truth
## Prediction    0    1
##          0 2023   20
##          1  125 8427
```

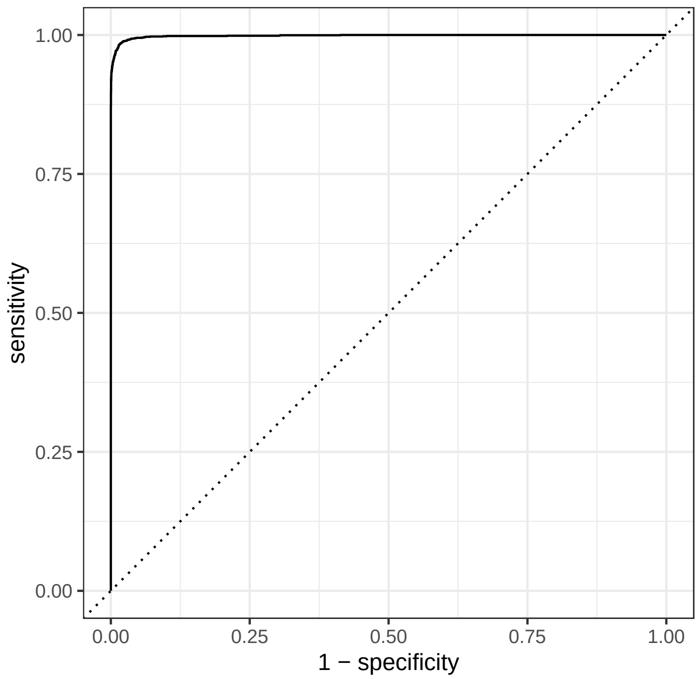
```r
pred_ranger %>%
  metrics(truth = loan_status_level, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.986
## 2 kap      binary         0.957
```

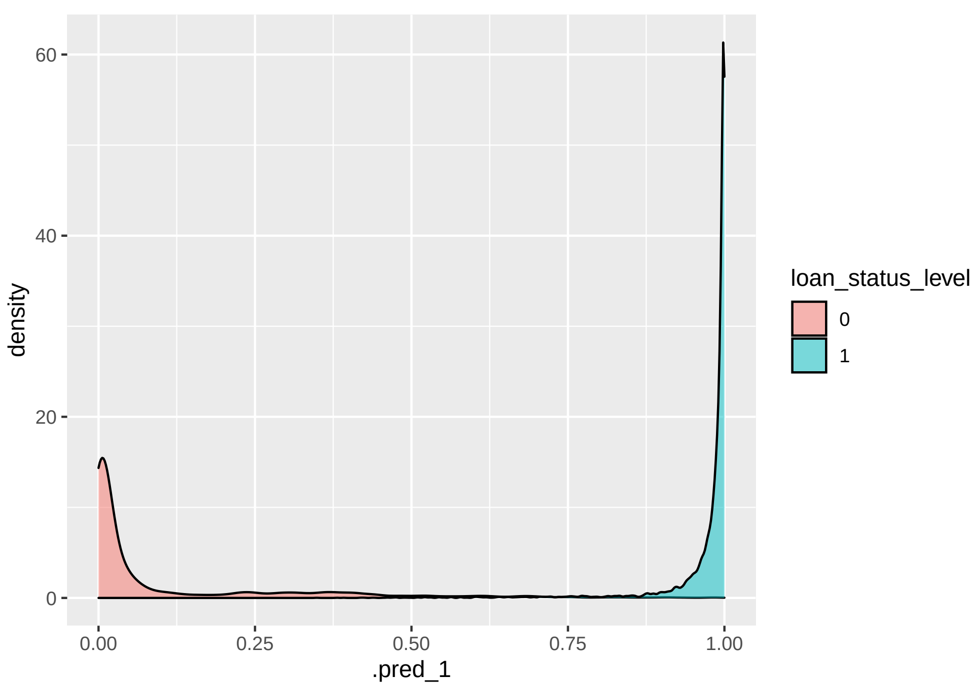
```r
# plot
plot_ranger <- mod_ranger %>%
  predict(samp_testing, type = "prob") %>%
  bind_cols(samp_testing)

plot_ranger %>%
  roc_auc(loan_status_level, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.999
```

```r
plot_ranger %>%
  roc_curve(loan_status_level, .pred_0) %>%
  autoplot()
```




```
plot_ranger %>%
  ggplot() + geom_density(aes(x = .pred_1, fill = loan_status_level), alpha = 0.5)
```



Improving Model Performance

```r
rf_model <- rand_forest(trees = tune()) %>%
  set_engine("ranger") %>%
  set_mode("classification")

rf_wflow <- workflow() %>%
  add_recipe(samp_recipe) %>%
  add_model(rf_model)

folds <- vfold_cv(training(samp_split), v = 10)

rf_grid <- expand.grid(trees = seq(50,800, by = 50))

rf_tune_resultes <- rf_wflow %>%
  tune_grid(resamples = folds, grid = rf_grid)

#rf_tune_resultes %>%
  #collect_metrics()

rf_trees <- rf_tune_resultes %>%
  select_best("accuracy")

rf_trees
```

```
## # A tibble: 1 x 2
##   trees .config
##   <dbl> <chr>
## 1   750 Preprocessor1_Model15
```

```r
rf_acc <- rf_wflow %>%
  finalize_workflow(rf_trees) %>%
  last_fit(samp_split) %>%
  collect_metrics()

rf_acc
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy binary         0.987 Preprocessor1_Model1
## 2 roc_auc  binary         0.999 Preprocessor1_Model1
```

Model 4: Rpart

Setup the Model

```r
mod_rp <- decision_tree(cost_complexity = 0.001, tree_depth = 6) %>%
  set_engine("rpart") %>%
  set_mode("classification") %>%
  fit(loan_status_level ~ ., data = samp_training)

mod_rp
```

```
## parsnip model object
##
## Fit time:  4.2s
## n= 31786
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 31786 6212 1 (0.195431951 0.804568049)
##     2) recoveries>=0.01 4329    0 0 (1.000000000 0.000000000) *
##     3) recoveries< 0.01 27457 1883 1 (0.068579961 0.931420039)
##       6) last_pymnt_amnt< 950.505 11383 1840 1 (0.161644558 0.838355442)
##        12) term=60 months 2366  985 0 (0.583685545 0.416314455)
##          24) initial_list_status=w 1239  283 0 (0.771589992 0.228410008)
##            48) last_pymnt_amnt>=204.66 1135  188 0 (0.834361233 0.165638767)
##              96) total_rec_prncp< 34586.38 1118  171 0 (0.847048301 0.152951699) *
##              97) total_rec_prncp>=34586.38 17    0 1 (0.000000000 1.000000000) *
##            49) last_pymnt_amnt< 204.66 104    9 1 (0.086538462 0.913461538) *
##          25) initial_list_status=f 1127  425 1 (0.377107365 0.622892635)
##            50) total_rec_prncp< 9998.06 197   37 0 (0.812182741 0.187817259)
##             100) loan_amnt>=9962.5 157    0 0 (1.000000000 0.000000000) *
##             101) loan_amnt< 9962.5 40    3 1 (0.075000000 0.925000000) *
##            51) total_rec_prncp>=9998.06 930  265 1 (0.284946237 0.715053763) *
##        13) term=36 months 9017  459 1 (0.050903848 0.949096152)
##          26) total_rec_prncp< 998.95 61    0 0 (1.000000000 0.000000000) *
##          27) total_rec_prncp>=998.95 8956  398 1 (0.044439482 0.955560518) *
##       7) last_pymnt_amnt>=950.505 16074   43 1 (0.002675128 0.997324872) *
```

Evaluating Model Performance

```r
# prediction on testing data set
pred_rp <- mod_rp %>%
  predict(samp_testing) %>%
  bind_cols(samp_testing)

pred_rp %>%
  conf_mat(truth = loan_status_level, estimate = .pred_class)
```

```
##           Truth
## Prediction    0    1
##          0 1886   64
##          1  262 8383
```

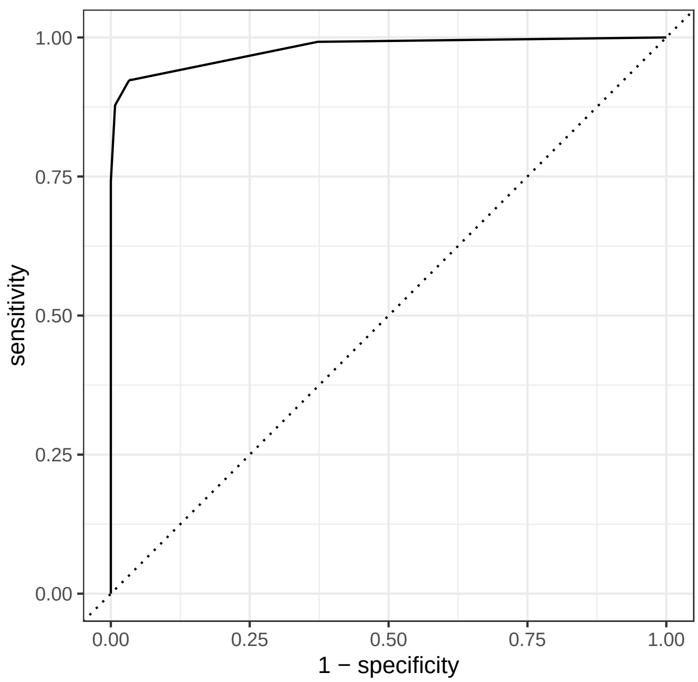
```r
pred_rp %>%
  metrics(truth = loan_status_level, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.969
## 2 kap      binary         0.901
```

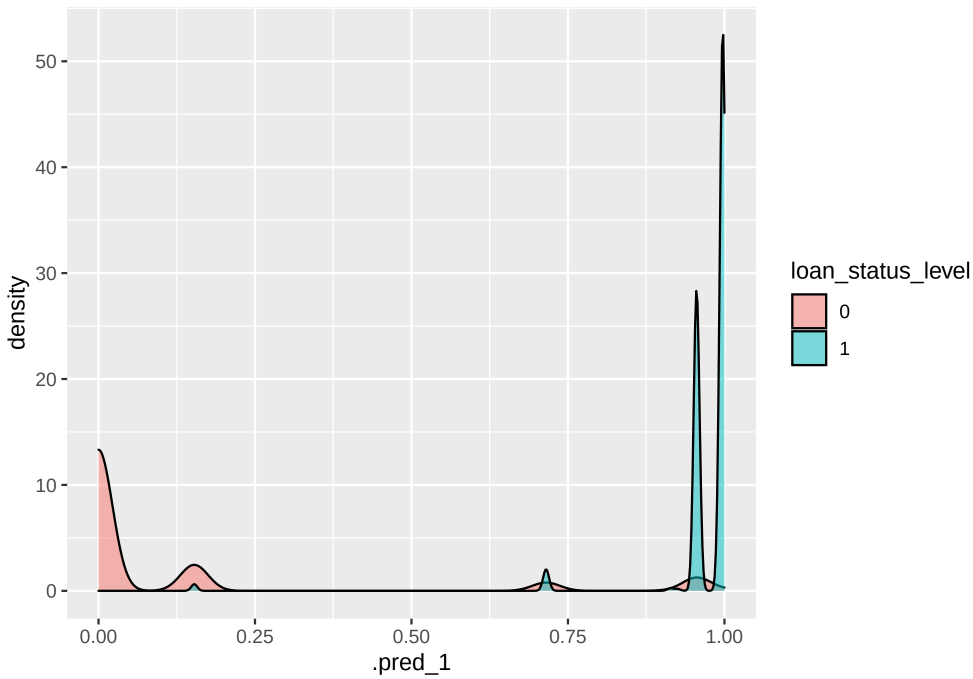
```r
# plot
plot_rp <- mod_rp %>%
  predict(samp_testing, type = "prob") %>%
  bind_cols(samp_testing)

plot_rp %>%
  roc_auc(loan_status_level, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.979
```

```r
plot_rp %>%
  roc_curve(loan_status_level, .pred_0) %>%
  autoplot()
```




```r
plot_rp %>%
  ggplot() + geom_density(aes(x = .pred_1, fill = loan_status_level), alpha = 0.5)
```



Improving Model Performance

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

rp_wflow <- workflow() %>%
  add_recipe(samp_recipe) %>%
  add_model(rp_model)

folds <- vfold_cv(training(samp_split), v = 10)

rp_grid <- grid_regular(cost_complexity(), tree_depth(), levels = 5)

rp_tune_resultes <- rp_wflow %>%
  tune_grid(resamples = folds, grid = rp_grid)

#rp_tune_resultes %>%
  #collect_metrics()

rp_trees <- rp_tune_resultes %>%
  select_best("accuracy")

rp_trees
```

```
## # A tibble: 1 x 3
##   cost_complexity tree_depth .config
##             <dbl>      <int> <chr>
## 1        0.000562         15 Preprocessor1_Model24
```

```
rp_acc <- rp_wflow %>%
  finalize_workflow(rp_trees) %>%
  last_fit(samp_split) %>%
  collect_metrics()

rp_acc
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy binary         0.984 Preprocessor1_Model1
## 2 roc_auc  binary         0.989 Preprocessor1_Model1
```

Model 5: XGBoost

Setup the Model

```
mod_xgb <- boost_tree(trees = 300) %>%
  set_engine("xgboost") %>%
  set_mode("classification") %>%
  fit(loan_status_level ~ ., data = samp_training)
```

```
## [01:19:42] WARNING: amalgamation/../src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evalu
```

```
# mod_xgb
```

Evaluating Model Performance

```
# prediction on testing data set
pred_xgb <- mod_xgb %>%
  predict(samp_testing) %>%
  bind_cols(samp_testing)

pred_xgb %>%
  conf_mat(truth = loan_status_level, estimate = .pred_class)
```

```
##           Truth
## Prediction    0    1
##          0 2103    3
##          1   45 8444
```

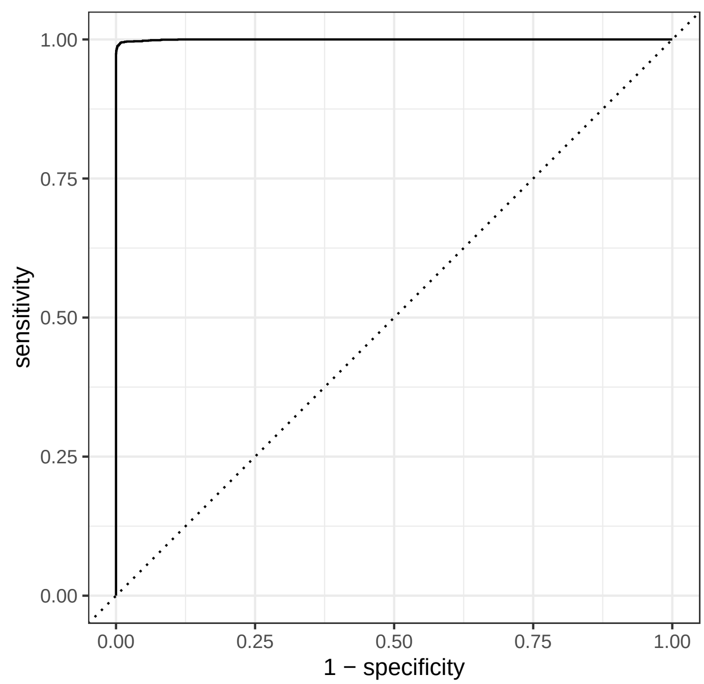
```
pred_xgb %>%
  metrics(truth = loan_status_level, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.995
## 2 kap      binary         0.986
```

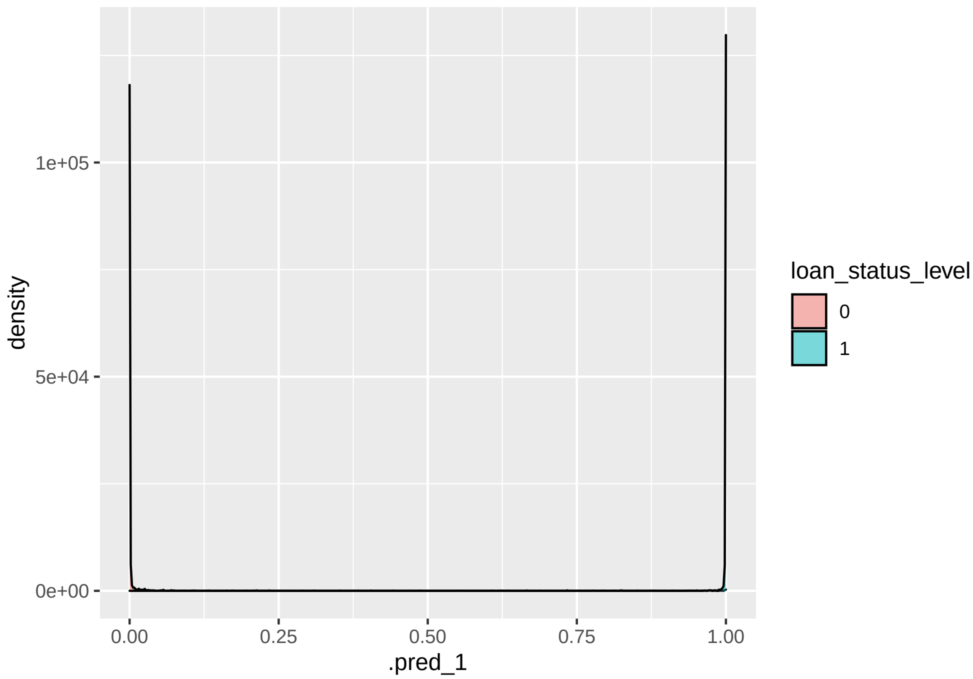
```r
# plot
plot_xgb <- mod_xgb %>%
  predict(samp_testing, type = "prob") %>%
  bind_cols(samp_testing)

plot_xgb %>%
  roc_auc(loan_status_level, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary          1.00
```

```r
plot_xgb %>%
  roc_curve(loan_status_level, .pred_0) %>%
  autoplot()
```




```r
plot_xgb %>%
  ggplot() + geom_density(aes(x = .pred_1, fill = loan_status_level), alpha = 0.5)
```



Model 6: Logistic Regression using Regularization

Setup the Model

```
mod_glm <- logistic_reg(penalty = 0.001, mixture = 0.5) %>%
  set_engine("glmnet") %>%
  set_mode("classification") %>%
  fit(loan_status_level ~ ., data = samp_training)

tidy(mod_glm)
```

```
## # A tibble: 103 x 3
##    term            estimate penalty
##    <chr>              <dbl>   <dbl>
##  1 (Intercept)     -4.91      0.001
##  2 loan_amnt       -0.000292  0.001
##  3 funded_amnt     -0.000289  0.001
##  4 funded_amnt_inv -0.000284  0.001
##  5 term60 months   -2.45      0.001
##  6 int_rate         0.0581    0.001
##  7 installment     -0.000730  0.001
##  8 gradeB           0         0.001
##  9 gradeC          -0.0513    0.001
## 10 gradeD          -0.181     0.001
## # ... with 93 more rows
```

Evaluating Model Performance

```r
# prediction on testing data set
pred_glm <- mod_glm %>%
  predict(samp_testing) %>%
  bind_cols(samp_testing)

pred_glm %>%
  conf_mat(truth = loan_status_level, estimate = .pred_class)
```

```
##           Truth
## Prediction    0    1
##          0 1975   17
##          1  173 8430
```

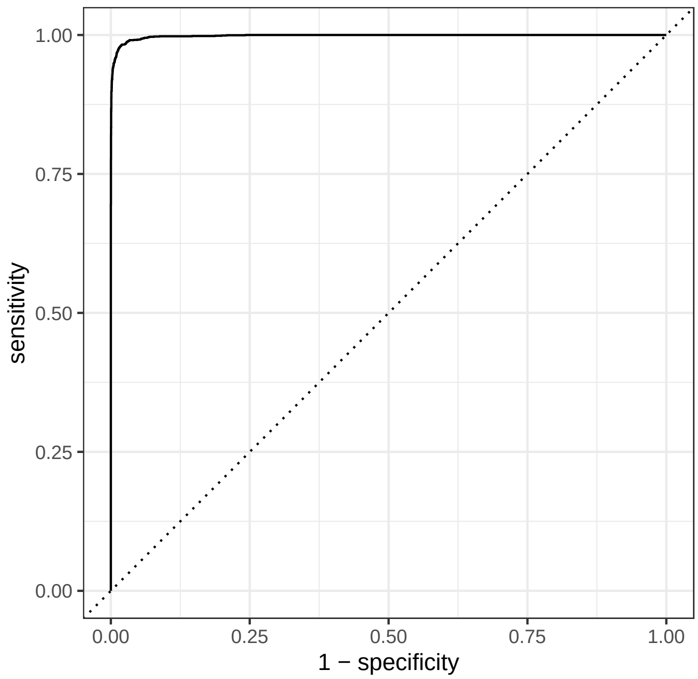
```r
pred_glm %>%
  metrics(truth = loan_status_level, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.982
## 2 kap      binary         0.943
```

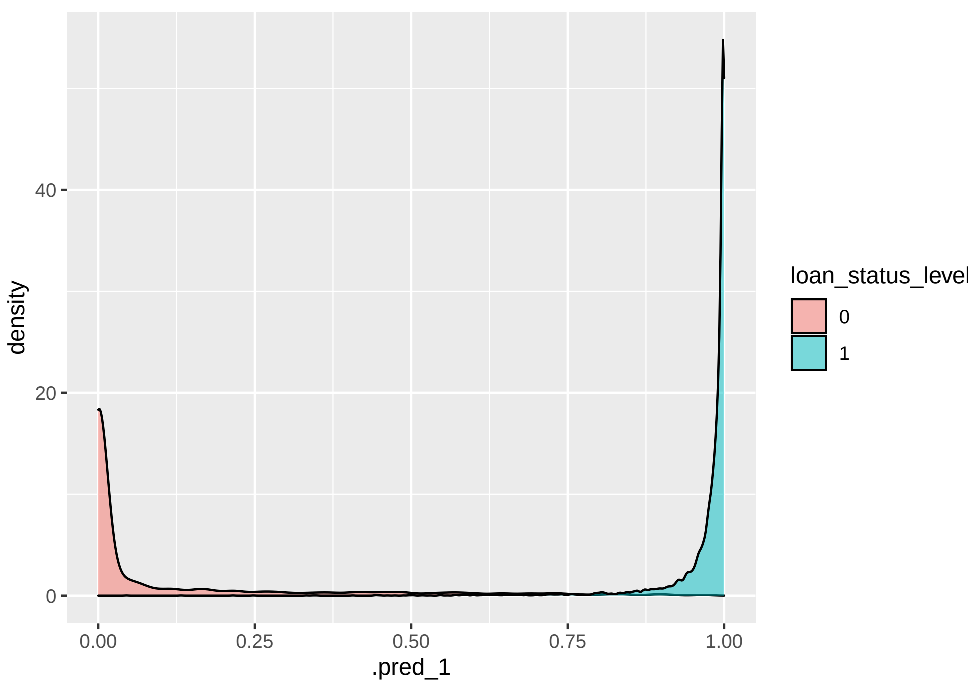
```r
# plot
plot_glm <- mod_glm %>%
  predict(samp_testing, type = "prob") %>%
  bind_cols(samp_testing)

plot_glm %>%
  roc_auc(loan_status_level, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.998
```

```r
plot_glm %>%
  roc_curve(loan_status_level, .pred_0) %>%
  autoplot()
```




```
plot_glm %>%
  ggplot() + geom_density(aes(x = .pred_1, fill = loan_status_level), alpha = 0.5)
```



Model 7: Naive Bayes

Setup the Model

```r
mod_nb <- naive_Bayes(Laplace = 1) %>%
  set_engine("klaR") %>%
  set_mode("classification") %>%
  fit(loan_status_level ~ ., data = samp_training)

# mod_nb
```

Evaluating Model Performance

```r
# prediction on testing data set
pred_nb <- mod_nb %>%
  predict(samp_testing) %>%
  bind_cols(samp_testing)

pred_nb %>%
  conf_mat(truth = loan_status_level, estimate = .pred_class)
```

```
##           Truth
## Prediction    0    1
##          0   87    1
##          1 2061 8446
```

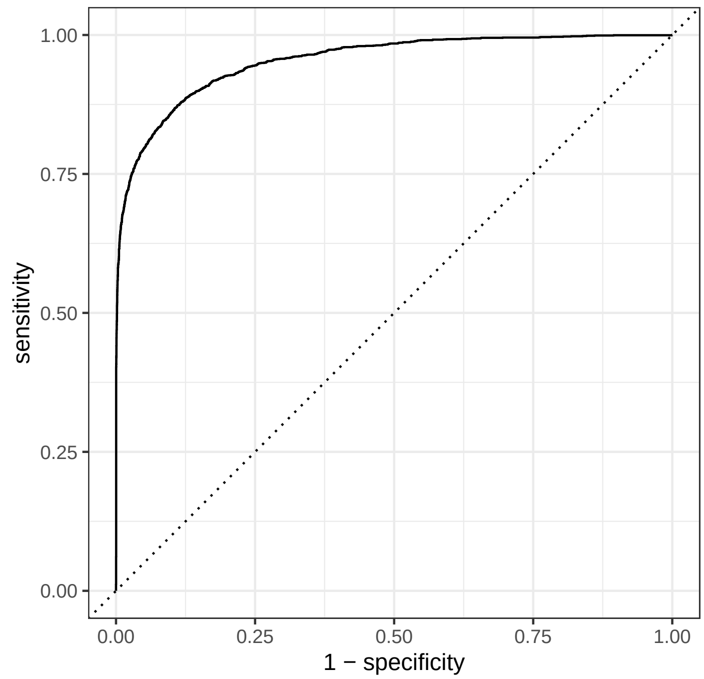
```r
pred_nb %>%
  metrics(truth = loan_status_level, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary        0.805
## 2 kap      binary        0.0629
```

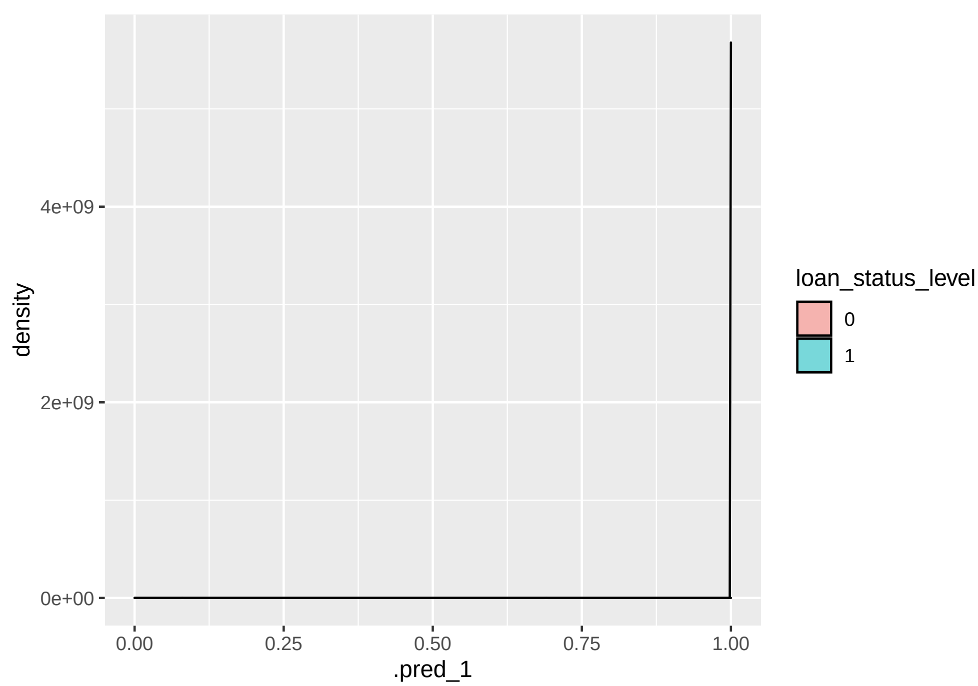
```r
# plot
plot_nb <- mod_nb %>%
  predict(samp_testing, type = "prob") %>%
  bind_cols(samp_testing)

plot_nb %>%
  roc_auc(loan_status_level, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.954
```

```r
plot_nb %>%
  roc_curve(loan_status_level, .pred_0) %>%
  autoplot()
```




```
plot_nb %>%
  ggplot() + geom_density(aes(x = .pred_1, fill = loan_status_level), alpha = 0.5)
```



Conclusion

Since the data set contains some nominal predictors (categorical variables), we can't improve some models' performance, which means some errors exist when processing tuning, such as Boosted C50 and XGBoost. Our tuning improves a little bit on some models' performance.

The following table shows the statistic of accuracy and AUC for all models:

.estimator	model	accuracy	kap
binary	XGBoost	0.9954696	0.9858826
binary	Boosted C50	0.9933931	0.9793906
binary	Random Forest	0.9863143	0.9568788
binary	glm	0.9820670	0.9429822
binary	Rpart	0.9692308	0.9014309
binary	kNN	0.9009910	0.6412272
binary	Naive Bayes	0.8053799	0.0628628
binary	Null	0.7972629	0.0000000

.metric	.estimator	.estimate	model
roc_auc	binary	0.9996774	XGBoost
roc_auc	binary	0.9995208	Boosted C50
roc_auc	binary	0.9985288	Random Forest
roc_auc	binary	0.9983066	glm
roc_auc	binary	0.9791188	Rpart
roc_auc	binary	0.9537031	Naive Bayes
roc_auc	binary	0.9187735	kNN
roc_auc	binary	0.5000000	Null

Among these 8 models, xgboost model has the greatest statistic of accuracy, kappa and AUC. Its confusion matrix shows the number of true positives is 8444 and number of true negatives is 2103, where as the number of false negatives is 3 and number of false positives is 45. This is the best confusion matrix compared with other models.

As a result, in this case, we would select xgboost model is the best ML learning model is for classifying *Loan Status*.

Extra Credit

Subset the data for the years 2015.

```
# subset the data for the years 2015
club_2015 <- data %>%
  filter(str_detect(issue_d, '2015'))

# club_2015
```

Check if the data set contains any duplicate records.

```
# check duplicate data
get_dupes(club_2015)
```

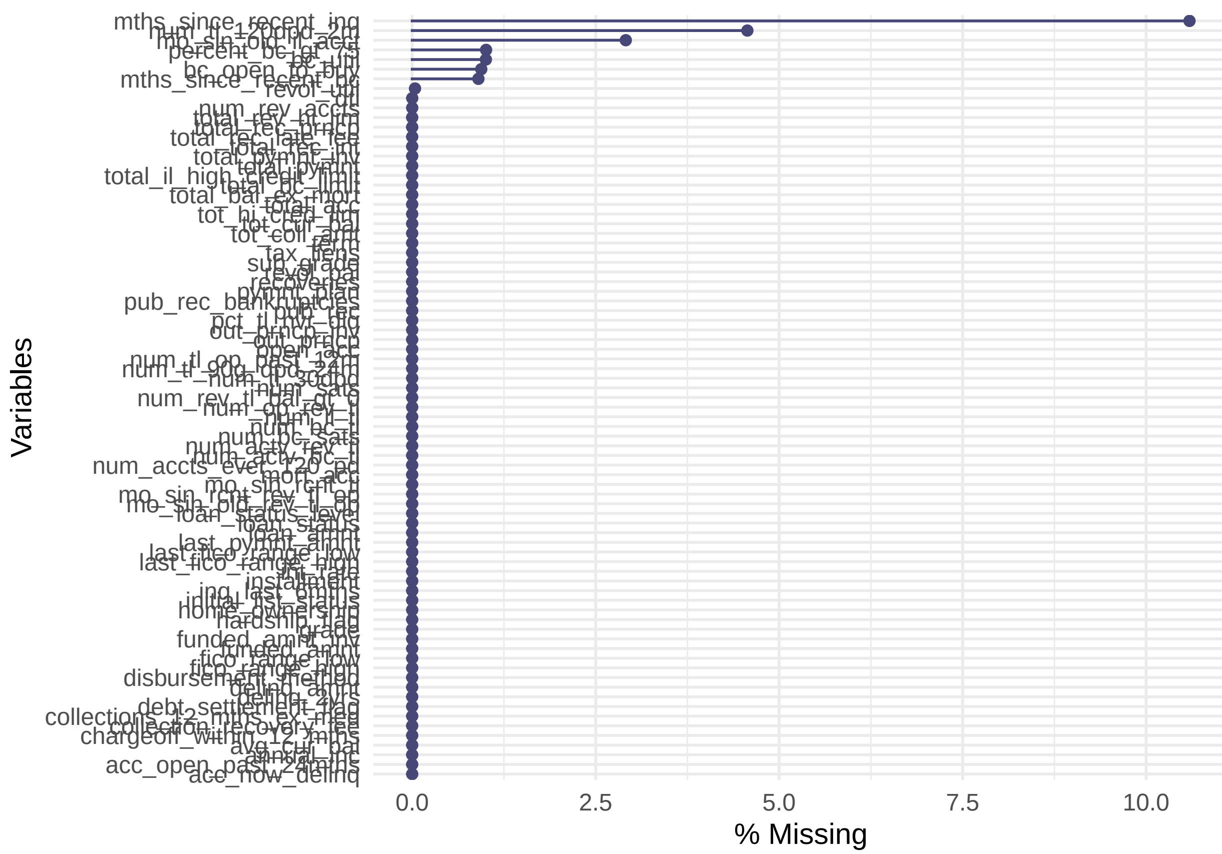
```
## No variable names specified - using all columns.

## No duplicate combinations found of: id, member_id, loan_amnt, funded_amnt, funded_amnt_inv, term, in

## Empty data.table (0 rows and 152 cols): id,member_id,loan_amnt,funded_amnt,funded_amnt_inv,term...
```

Select the variables which are the same as those in data from year 2012-2014.

```
var_2014 <- club_df %>%
  names()

club_df_2015 <- club_2015 %>%
  mutate_if(is.character, as.factor) %>%          # convert character vars to factor vars
  mutate(loan_status_level = ifelse(loan_status == "Fully Paid", 1,  0)) %>%   # classification
  mutate(loan_status_level = as.factor(loan_status_level)) %>%
  select(one_of(var_2014))

# head(club_df_2015)
```

Check missing values.

```
# check missing value
gg_miss_var(club_df_2015, show_pct = TRUE)
```



Summary of classification for the variable *loan_status*.

```
# check loan status level
club_df_2015 %>%
  group_by(loan_status, loan_status_level) %>%
  tally()
```

```
## # A tibble: 7 x 3
## # Groups:   loan_status [7]
##   loan_status        loan_status_level      n
##   <fct>              <fct>              <int>
## 1 Charged Off        0                  75803
## 2 Current            0                  43299
## 3 Default            0                      1
## 4 Fully Paid         1                 299742
## 5 In Grace Period    0                    612
## 6 Late (16-30 days)  0                    279
## 7 Late (31-120 days) 0                   1359
```

```
CrossTable(club_df_2015$loan_status_level, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
```

```
## |                         N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  421095
##
##
##           |         0 |         1 |
##           |-----------|-----------|
##           |    121353 |    299742 |
##           |     0.288 |     0.712 |
##           |-----------|-----------|
##
##
##
##
```

Sample 10% of data set for testing

```
# sample 10% of data set
set.seed(999)
club_samp_2015 <- club_df_2015 %>%
  slice_sample(n = 0.1*nrow(club_2015))
```

Summary of classification for the variable *loan_status*.

```
#summarize the y-variable
CrossTable(club_samp_2015$loan_status_level, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                         N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  42109
##
##
##           |         0 |         1 |
##           |-----------|-----------|
##           |     12013 |     30096 |
##           |     0.285 |     0.715 |
##           |-----------|-----------|
##
##
##
##
```

Because there are different levels in *home_ownership* between 2 data sets, we need to drop levels in order to make 2 data sets correspond.

```
levels(club_df$home_ownership)
```

```
## [1] "ANY"      "MORTGAGE" "NONE"     "OTHER"    "OWN"      "RENT"
```

```
levels(club_samp_2015$home_ownership)
```

```
## [1] "ANY"      "MORTGAGE" "OWN"      "RENT"
```

```
club_re <- club_df %>%
  mutate(home_ownership = as.character(home_ownership)) %>%
  filter(home_ownership == "ANY" | home_ownership == "MORTGAGE" |
         home_ownership == "OWN" | home_ownership == "RENT") %>%
  mutate_if(is.character, as.factor)

levels(club_re$home_ownership)
```

```
## [1] "ANY"      "MORTGAGE" "OWN"      "RENT"
```

Create Training nad Testing data set.

```
set.seed(999)
club_samp <- club_re %>%
  slice_sample(n = 0.1*nrow(club))
```

```
set.seed(999)
samp_split <- club_samp %>%
  initial_split(prop = 0.75)


samp_recipe <- training(samp_split) %>%
  recipe(loan_status_level ~ .) %>%
  step_rm(loan_status) %>%
  step_nzv(all_predictors()) %>%
  step_knnimpute(all_predictors()) %>%
  prep()

summary(samp_recipe)
```

```
## # A tibble: 61 x 4
##    variable        type    role      source
##    <chr>           <chr>   <chr>     <chr>
##  1 loan_amnt       numeric predictor original
##  2 funded_amnt     numeric predictor original
##  3 funded_amnt_inv numeric predictor original
##  4 term            nominal predictor original
##  5 int_rate        numeric predictor original
##  6 installment     numeric predictor original
##  7 grade           nominal predictor original
##  8 sub_grade       nominal predictor original
##  9 home_ownership  nominal predictor original
## 10 annual_inc      numeric predictor original
## # ... with 51 more rows
```

```
tidy(samp_recipe)
```

```
## # A tibble: 3 x 6
##   number operation type      trained skip  id
##    <int> <chr>     <chr>     <lgl>   <lgl> <chr>
## 1      1 step      rm        TRUE    FALSE rm_XPPN6
## 2      2 step      nzv       TRUE    FALSE nzv_0Xq94
## 3      3 step      knnimpute TRUE    FALSE knnimpute_5S9Hw
```

```
samp_testing_2015 <- samp_recipe %>%
  bake(club_samp_2015)

samp_testing_2015
```

```
## # A tibble: 42,109 x 61
##    loan_amnt funded_amnt funded_amnt_inv term      int_rate installment grade
##        <dbl>       <dbl>           <dbl> <fct>        <dbl>       <dbl> <fct>
##  1     30000       30000           30000 60 months    14.6        708.  C
##  2     15600       15600           15600 36 months     9.99       503.  B
##  3      8000        8000            8000 36 months    12.0        266.  C
##  4     20675       20675           20675 60 months    19.5        542.  E
##  5      6000        6000            6000 36 months    17.0        214.  D
##  6     22000       22000           21850 60 months    23.0        620.  F
##  7     25200       25200           25200 60 months     7.89       510.  A
##  8     10000       10000           10000 60 months    12.0        222.  B
##  9      1800        1800            1800 36 months    12.7         60.4 C
## 10      6000        6000            6000 36 months     7.89       188.  A
## # ... with 42,099 more rows, and 54 more variables: sub_grade <fct>,
## #   home_ownership <fct>, annual_inc <dbl>, dti <dbl>, delinq_2yrs <dbl>,
## #   fico_range_low <dbl>, fico_range_high <dbl>, inq_last_6mths <dbl>,
## #   open_acc <dbl>, pub_rec <dbl>, revol_bal <dbl>, revol_util <dbl>,
## #   total_acc <dbl>, initial_list_status <fct>, total_pymnt <dbl>,
## #   total_pymnt_inv <dbl>, total_rec_prncp <dbl>, total_rec_int <dbl>,
## #   recoveries <dbl>, collection_recovery_fee <dbl>, last_pymnt_amnt <dbl>,
## #   last_fico_range_high <dbl>, last_fico_range_low <dbl>, tot_cur_bal <dbl>,
## #   total_rev_hi_lim <dbl>, acc_open_past_24mths <dbl>, avg_cur_bal <dbl>,
## #   bc_open_to_buy <dbl>, bc_util <dbl>, mo_sin_old_il_acct <dbl>,
## #   mo_sin_old_rev_tl_op <dbl>, mo_sin_rcnt_rev_tl_op <dbl>,
## #   mo_sin_rcnt_tl <dbl>, mort_acc <dbl>, mths_since_recent_bc <dbl>,
## #   mths_since_recent_inq <dbl>, num_accts_ever_120_pd <dbl>,
## #   num_actv_bc_tl <dbl>, num_actv_rev_tl <dbl>, num_bc_sats <dbl>,
## #   num_bc_tl <dbl>, num_il_tl <dbl>, num_op_rev_tl <dbl>, num_rev_accts <dbl>,
## #   num_rev_tl_bal_gt_0 <dbl>, num_sats <dbl>, num_tl_op_past_12m <dbl>,
## #   percent_bc_gt_75 <dbl>, pub_rec_bankruptcies <dbl>, tot_hi_cred_lim <dbl>,
## #   total_bal_ex_mort <dbl>, total_bc_limit <dbl>,
## #   total_il_high_credit_limit <dbl>, loan_status_level <fct>
```

```
samp_training <- juice(samp_recipe)

samp_training
```

```
## # A tibble: 31,786 x 61
```

```
##    loan_amnt funded_amnt funded_amnt_inv term      int_rate installment grade
##        <dbl>       <dbl>           <dbl> <fct>        <dbl>       <dbl> <fct>
##  1      5000        5000            5000 36 months    12.0         166. B
##  2     20000       20000           20000 36 months    21           754. E
##  3      6400        6400            6400 36 months     6.99        198. A
##  4     24250       24250           24250 60 months    11.4         533. B
##  5     23975       23975           23975 60 months    22.2         664. E
##  6      8000        8000            8000 36 months    12.5         268. B
##  7     15950       15950           15950 36 months    17.0         569. D
##  8     15000       15000           14950 60 months    16.2         366. C
##  9     15500       15500           15500 36 months    11.0         507. B
## 10     10000       10000           10000 60 months    17.0         248. D
## # ... with 31,776 more rows, and 54 more variables: sub_grade <fct>,
## #   home_ownership <fct>, annual_inc <dbl>, dti <dbl>, delinq_2yrs <dbl>,
## #   fico_range_low <dbl>, fico_range_high <dbl>, inq_last_6mths <dbl>,
## #   open_acc <dbl>, pub_rec <dbl>, revol_bal <dbl>, revol_util <dbl>,
## #   total_acc <dbl>, initial_list_status <fct>, total_pymnt <dbl>,
## #   total_pymnt_inv <dbl>, total_rec_prncp <dbl>, total_rec_int <dbl>,
## #   recoveries <dbl>, collection_recovery_fee <dbl>, last_pymnt_amnt <dbl>,
## #   last_fico_range_high <dbl>, last_fico_range_low <dbl>, tot_cur_bal <dbl>,
## #   total_rev_hi_lim <dbl>, acc_open_past_24mths <dbl>, avg_cur_bal <dbl>,
## #   bc_open_to_buy <dbl>, bc_util <dbl>, mo_sin_old_il_acct <dbl>,
## #   mo_sin_old_rev_tl_op <dbl>, mo_sin_rcnt_rev_tl_op <dbl>,
## #   mo_sin_rcnt_tl <dbl>, mort_acc <dbl>, mths_since_recent_bc <dbl>,
## #   mths_since_recent_inq <dbl>, num_accts_ever_120_pd <dbl>,
## #   num_actv_bc_tl <dbl>, num_actv_rev_tl <dbl>, num_bc_sats <dbl>,
## #   num_bc_tl <dbl>, num_il_tl <dbl>, num_op_rev_tl <dbl>, num_rev_accts <dbl>,
## #   num_rev_tl_bal_gt_0 <dbl>, num_sats <dbl>, num_tl_op_past_12m <dbl>,
## #   percent_bc_gt_75 <dbl>, pub_rec_bankruptcies <dbl>, tot_hi_cred_lim <dbl>,
## #   total_bal_ex_mort <dbl>, total_bc_limit <dbl>,
## #   total_il_high_credit_limit <dbl>, loan_status_level <fct>
```

Model: XGBoost

```
mod_xgb_2015 <- boost_tree(trees = 300) %>%
  set_engine("xgboost") %>%
  set_mode("classification") %>%
  fit(loan_status_level ~ ., data = samp_training)
```

Setup the Model

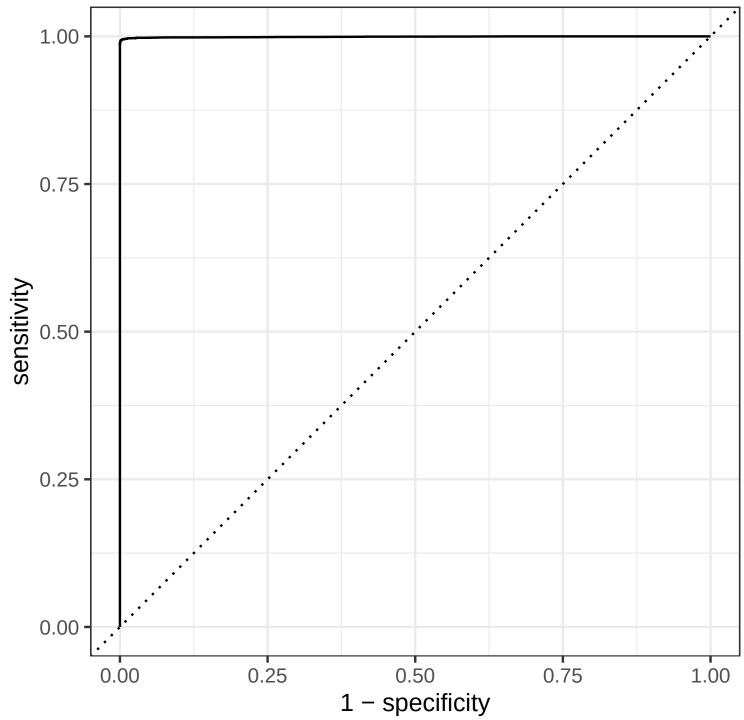
```
## [14:57:21] WARNING: amalgamation/../src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evalu
```

```
# mod_xgb_2015
```

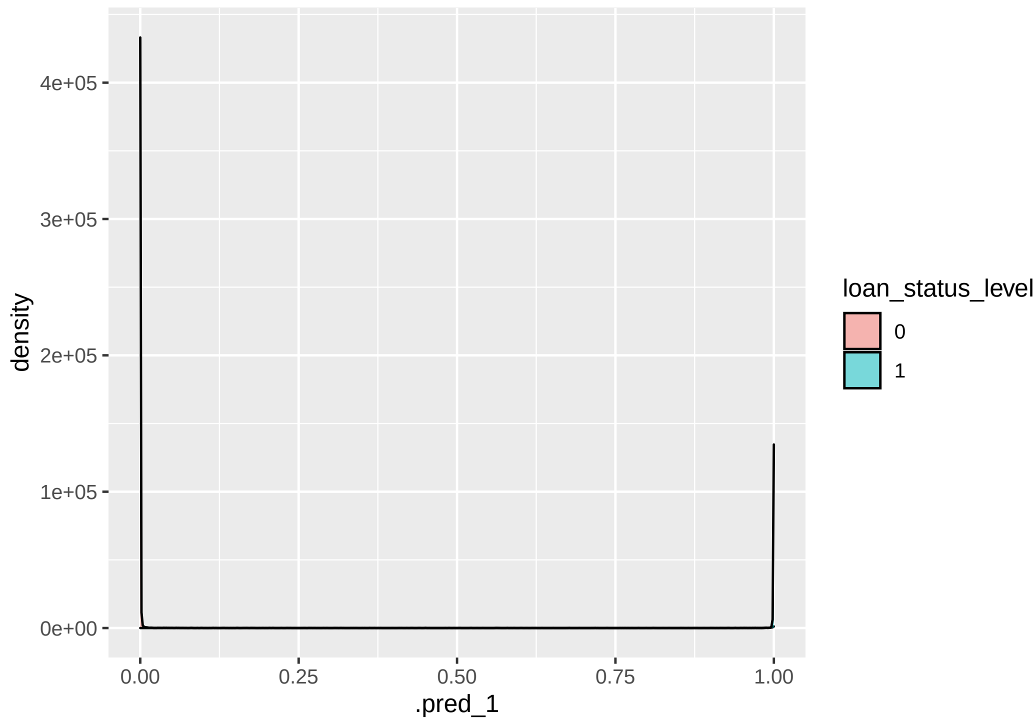
```
# plot
plot_xgb_2015 <- mod_xgb_2015 %>%
```

```
  predict(samp_testing_2015, type = "prob") %>%
  bind_cols(samp_testing_2015)

plot_xgb_2015 %>%
  roc_curve(loan_status_level, .pred_0) %>%
  autoplot()
```

Evaluating Model Performance




```
plot_xgb_2015 %>%
  ggplot() + geom_density(aes(x = .pred_1, fill = loan_status_level), alpha = 0.5)
```




```
# prediction on testing data set
pred_xgb_2015 <- mod_xgb_2015 %>%
  predict(samp_testing_2015) %>%
  bind_cols(samp_testing_2015)

pred_xgb_2015 %>%
  conf_mat(truth = loan_status_level, estimate = .pred_class)
```

```
##           Truth
## Prediction     0     1
##          0 11825     0
##          1   188 30096
```

```
pred_xgb_2015 %>%
  metrics(truth = loan_status_level, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.996
## 2 kap      binary         0.989
```

```
plot_xgb_2015 %>%
  roc_auc(loan_status_level, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.999
```