MIS 431 Summer 2023 Final Project

June 25th, 2023

**Name**: An Huynh

# Suppress dplyr summarise grouping warning messages  
options(dplyr.summarise.inform = FALSE)  
  
## Add R libraries here  
library(tidyverse)  
library(tidymodels)  
library(skimr)  
library(dplyr)  
library(ggplot2)  
library(cowplot)  
library(vip)  
library(rpart.plot)  
  
  
# Load data  
loans\_df <- read\_rds("/Users/AnhHuynh/Documents/SUMMER 2023/MIS 431/Final Project/loan\_data.rds")  
  
skim(loans\_df)

Data summary

|  |  |
| --- | --- |
| Name | loans\_df |
| Number of rows | 4110 |
| Number of columns | 16 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 8 |
| numeric | 8 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| --- | --- | --- | --- | --- | --- |
| loan\_default | 0 | 1 | FALSE | 2 | no: 2580, yes: 1530 |
| loan\_purpose | 0 | 1 | FALSE | 5 | deb: 1218, cre: 879, sma: 853, med: 635 |
| application\_type | 0 | 1 | FALSE | 2 | ind: 3494, joi: 616 |
| term | 0 | 1 | FALSE | 2 | thr: 2588, fiv: 1522 |
| homeownership | 0 | 1 | FALSE | 3 | mor: 1937, ren: 1666, own: 507 |
| missed\_payment\_2\_yr | 0 | 1 | FALSE | 2 | no: 3640, yes: 470 |
| history\_bankruptcy | 0 | 1 | FALSE | 2 | no: 3624, yes: 486 |
| history\_tax\_liens | 0 | 1 | FALSE | 2 | no: 4050, yes: 60 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| loan\_amount | 0 | 1 | 16692.79 | 10038.89 | 1000.00 | 9600.00 | 15000.00 | 24000.00 | 40000.00 | ▆▇▅▃▂ |
| installment | 0 | 1 | 489.42 | 289.50 | 31.04 | 274.82 | 421.97 | 663.98 | 1566.59 | ▇▇▅▂▁ |
| interest\_rate | 0 | 1 | 11.38 | 3.92 | 4.72 | 8.22 | 11.25 | 13.75 | 20.00 | ▆▆▇▃▃ |
| annual\_income | 0 | 1 | 73015.01 | 37203.11 | 3000.00 | 45000.00 | 65000.00 | 92000.00 | 200000.00 | ▃▇▃▁▁ |
| current\_job\_years | 0 | 1 | 5.80 | 3.69 | 0.00 | 2.00 | 5.00 | 10.00 | 10.00 | ▆▃▂▂▇ |
| debt\_to\_income | 0 | 1 | 20.04 | 14.23 | 0.00 | 11.85 | 18.59 | 26.13 | 437.61 | ▇▁▁▁▁ |
| total\_credit\_lines | 0 | 1 | 22.47 | 12.03 | 2.00 | 14.00 | 20.00 | 29.00 | 87.00 | ▇▇▂▁▁ |
| years\_credit\_history | 0 | 1 | 15.76 | 7.22 | 3.00 | 11.00 | 14.00 | 19.00 | 51.00 | ▆▇▂▁▁ |

# Data Analysis [30 Points]

## Question 1:

**Question**: Is annual income a factor of loan default?

**Answer**: Yes, both the summary table and the plot shows that lower income people tend to default on loans. On average, loan-default customers earn $8278.9 annually less than those who don’t default on loan. The 25th, 50th, and 75th percentiles of annual income reflected on the box plot yield same conclusion that those who don’t default on loans have higher income.

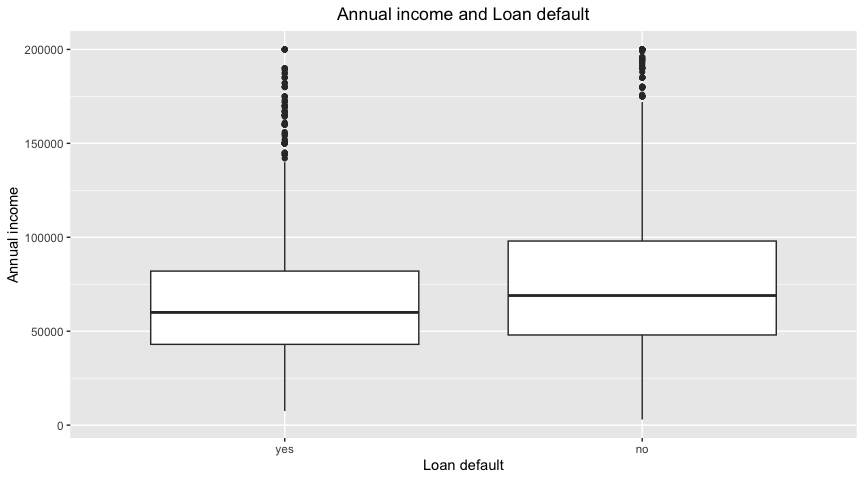
# Summary table

x1 <- loans\_df %>%  
 group\_by(loan\_default) %>%  
 summarize(n\_cust = n(),  
 avg\_income = mean(annual\_income),  
 med\_income = median(annual\_income))  
   
x1

## # A tibble: 2 × 4  
## loan\_default n\_cust avg\_income med\_income  
## <fct> <int> <dbl> <dbl>  
## 1 yes 1530 67819. 60000  
## 2 no 2580 76096. 69000

# Data visualization

ggplot(loans\_df, aes(x=loan\_default, y=annual\_income)) +  
 geom\_boxplot() +  
 labs(title = "Annual income and Loan default",  
 x="Loan default", y="Annual income") +  
 theme(plot.title = element\_text(hjust = 0.5))



## Question 2

**Question**: How do installment and interest rate impact loan default?

**Answer**: Both installment and interest rate are factors explaining loan default, and interest rate is a stronger indicator. On average, those who default on loans have higher percent interest rate, about 5.6%, and pay roughly $73 more each month.

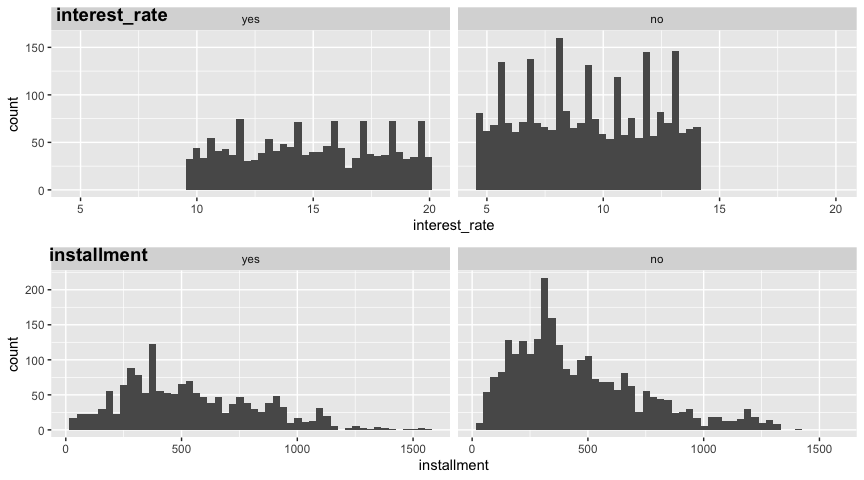
#Summary Table

x2 <- loans\_df %>%  
 group\_by(loan\_default) %>%  
 summarize(avg\_interest = mean(interest\_rate),  
 avg\_installment = mean(installment))  
x2

## # A tibble: 2 × 3  
## loan\_default avg\_interest avg\_installment  
## <fct> <dbl> <dbl>  
## 1 yes 14.9 535.  
## 2 no 9.30 462.

# Data visualization

plot1 <- ggplot(loans\_df, aes(x=interest\_rate))+  
 geom\_histogram(bins = 50 ) +  
 facet\_wrap(~loan\_default)  
  
plot2 <- ggplot(loans\_df,aes(x=installment)) +  
 geom\_histogram(bins=50) +  
 facet\_wrap(~loan\_default)  
  
plot\_grid(plot1, plot2, labels=c("interest\_rate", "installment"), nrow=2 )



## Question 3

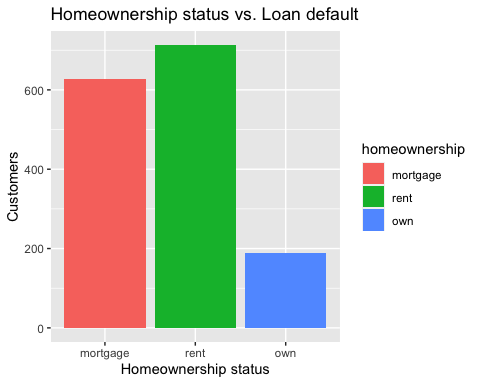
**Question**: Is there a difference among homeownership status regard to loan default?

**Answer**: Yes, there is a striking difference among the three groups. Renters make up the majority of those who have loan default, followed by those who are still contributing to home mortgage, and those who have paid off their mortgage account for the lowest percent.

x3 <- loans\_df %>%  
 filter(loan\_default=="yes") %>%  
 group\_by(homeownership) %>%  
 summarize(n\_customers = n()) %>%  
 arrange(homeownership)  
x3

## # A tibble: 3 × 2  
## homeownership n\_customers  
## <fct> <int>  
## 1 mortgage 628  
## 2 rent 713  
## 3 own 189

ggplot(x3, aes(x=homeownership, y=n\_customers, fill=homeownership)) +  
 geom\_bar(position = "dodge", stat = "identity") +  
 labs (title = "Homeownership status vs. Loan default",  
 x="Homeownership status", y="Customers")



# Predictive Modeling [70 Points]

## Model 1

### Data Splitting

set.seed(345)   
  
loan\_split <- initial\_split(loans\_df, prop=3/4, strata = loan\_default)  
  
#Training set  
loan\_train <- loan\_split %>% training()  
  
#Testing set  
loan\_test <- loan\_split %>% testing()

### Featuring Engineer

loan\_recipe <- recipe(loan\_default ~ ., data = loan\_train) %>%   
 step\_YeoJohnson(all\_numeric(), -all\_outcomes()) %>%   
 step\_normalize(all\_numeric(), -all\_outcomes()) %>%   
 step\_dummy(all\_nominal(), -all\_outcomes())

loan\_recipe %>%   
 prep() %>%   
 bake(new\_data = loan\_train)

## # A tibble: 3,082 × 20  
## loan\_amount installment interest\_rate annual\_income current\_job\_years  
## <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1.16 1.42 -0.551 1.89 1.10   
## 2 -1.62 -1.70 -0.345 0.189 -0.377  
## 3 -1.23 -1.22 -0.838 0.133 -0.377  
## 4 0.0145 0.0572 0.0439 -0.828 1.10   
## 5 1.64 0.990 -1.06 2.22 1.10   
## 6 -1.14 -1.10 0.0439 -1.36 1.10   
## 7 -0.369 -0.841 -0.912 -1.18 -0.104  
## 8 -0.554 -0.547 -1.06 -0.166 1.10   
## 9 -1.06 -1.05 -0.147 -0.437 -0.668  
## 10 -0.554 -0.518 -0.692 0.522 1.10   
## # ℹ 3,072 more rows  
## # ℹ 15 more variables: debt\_to\_income <dbl>, total\_credit\_lines <dbl>,  
## # years\_credit\_history <dbl>, loan\_default <fct>,  
## # loan\_purpose\_credit\_card <dbl>, loan\_purpose\_medical <dbl>,  
## # loan\_purpose\_small\_business <dbl>, loan\_purpose\_home\_improvement <dbl>,  
## # application\_type\_joint <dbl>, term\_five\_year <dbl>,  
## # homeownership\_rent <dbl>, homeownership\_own <dbl>, …

### Model Specification

logistic\_model <- logistic\_reg() %>%   
 set\_engine('glm') %>%   
 set\_mode('classification')  
logistic\_model

## Logistic Regression Model Specification (classification)  
##   
## Computational engine: glm

### Create a workflow

loan\_wf <- workflow() %>%  
 add\_model(logistic\_model) %>%  
 add\_recipe(loan\_recipe)  
loan\_wf

## ══ Workflow ════════════════════════════════════════════════════════════════════  
## Preprocessor: Recipe  
## Model: logistic\_reg()  
##   
## ── Preprocessor ────────────────────────────────────────────────────────────────  
## 3 Recipe Steps  
##   
## • step\_YeoJohnson()  
## • step\_normalize()  
## • step\_dummy()  
##   
## ── Model ───────────────────────────────────────────────────────────────────────  
## Logistic Regression Model Specification (classification)  
##   
## Computational engine: glm

### Fit the model

loan\_logistic\_fit <- loan\_wf %>%  
 fit(data=loan\_train)  
  
loan\_logistic\_fit

## ══ Workflow [trained] ══════════════════════════════════════════════════════════  
## Preprocessor: Recipe  
## Model: logistic\_reg()  
##   
## ── Preprocessor ────────────────────────────────────────────────────────────────  
## 3 Recipe Steps  
##   
## • step\_YeoJohnson()  
## • step\_normalize()  
## • step\_dummy()  
##   
## ── Model ───────────────────────────────────────────────────────────────────────  
##   
## Call: stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Coefficients:  
## (Intercept) loan\_amount   
## 8.79197 28.02629   
## installment interest\_rate   
## -25.22909 -2.84596   
## annual\_income current\_job\_years   
## -0.16749 0.13550   
## debt\_to\_income total\_credit\_lines   
## -0.28770 0.24616   
## years\_credit\_history loan\_purpose\_credit\_card   
## -0.04054 -1.18667   
## loan\_purpose\_medical loan\_purpose\_small\_business   
## -1.75043 0.11032   
## loan\_purpose\_home\_improvement application\_type\_joint   
## -0.02612 -0.56007   
## term\_five\_year homeownership\_rent   
## -16.36598 -0.81587   
## homeownership\_own missed\_payment\_2\_yr\_no   
## -0.58275 0.64769   
## history\_bankruptcy\_no history\_tax\_liens\_no   
## -0.18180 -0.34375   
##   
## Degrees of Freedom: 3081 Total (i.e. Null); 3062 Residual  
## Null Deviance: 4069   
## Residual Deviance: 756.1 AIC: 796.1

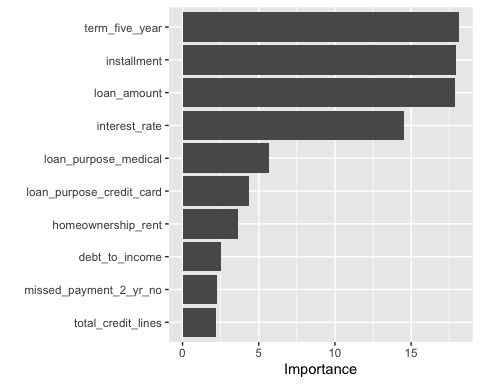
### Exploring trained model

#Extract train model from workflow  
loan\_trained\_model <- loan\_logistic\_fit %>%  
 extract\_fit\_parsnip()  
loan\_trained\_model

## parsnip model object  
##   
##   
## Call: stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Coefficients:  
## (Intercept) loan\_amount   
## 8.79197 28.02629   
## installment interest\_rate   
## -25.22909 -2.84596   
## annual\_income current\_job\_years   
## -0.16749 0.13550   
## debt\_to\_income total\_credit\_lines   
## -0.28770 0.24616   
## years\_credit\_history loan\_purpose\_credit\_card   
## -0.04054 -1.18667   
## loan\_purpose\_medical loan\_purpose\_small\_business   
## -1.75043 0.11032   
## loan\_purpose\_home\_improvement application\_type\_joint   
## -0.02612 -0.56007   
## term\_five\_year homeownership\_rent   
## -16.36598 -0.81587   
## homeownership\_own missed\_payment\_2\_yr\_no   
## -0.58275 0.64769   
## history\_bankruptcy\_no history\_tax\_liens\_no   
## -0.18180 -0.34375   
##   
## Degrees of Freedom: 3081 Total (i.e. Null); 3062 Residual  
## Null Deviance: 4069   
## Residual Deviance: 756.1 AIC: 796.1

### Variable Importance

vip(loan\_trained\_model)

 ### Evaluate Performance

#Predicting categories  
pred\_categories <- predict(loan\_logistic\_fit,new\_data = loan\_test)  
pred\_categories

## # A tibble: 1,028 × 1  
## .pred\_class  
## <fct>   
## 1 yes   
## 2 yes   
## 3 yes   
## 4 no   
## 5 no   
## 6 yes   
## 7 yes   
## 8 yes   
## 9 yes   
## 10 no   
## # ℹ 1,018 more rows

#Predicting probabilities  
pred\_prob <- predict(loan\_logistic\_fit,new\_data = loan\_test,type = 'prob')  
pred\_prob

## # A tibble: 1,028 × 2  
## .pred\_yes .pred\_no  
## <dbl> <dbl>  
## 1 0.967 0.0334   
## 2 0.637 0.363   
## 3 1.00 0.0000000468  
## 4 0.0180 0.982   
## 5 0.000107 1.00   
## 6 0.848 0.152   
## 7 0.652 0.348   
## 8 0.999 0.000723   
## 9 0.642 0.358   
## 10 0.0903 0.910   
## # ℹ 1,018 more rows

#Combine results from above with true response variable values in our test data set  
test\_results <- loan\_test %>%  
 select(loan\_default) %>%  
 bind\_cols(pred\_categories) %>%  
 bind\_cols(pred\_prob)  
test\_results

## # A tibble: 1,028 × 4  
## loan\_default .pred\_class .pred\_yes .pred\_no  
## <fct> <fct> <dbl> <dbl>  
## 1 yes yes 0.967 0.0334   
## 2 yes yes 0.637 0.363   
## 3 yes yes 1.00 0.0000000468  
## 4 no no 0.0180 0.982   
## 5 no no 0.000107 1.00   
## 6 no yes 0.848 0.152   
## 7 yes yes 0.652 0.348   
## 8 yes yes 0.999 0.000723   
## 9 yes yes 0.642 0.358   
## 10 no no 0.0903 0.910   
## # ℹ 1,018 more rows

### Exploring Performance Metrics

#### Confusion Matrix

conf\_mat(test\_results,truth = loan\_default,estimate = .pred\_class)

## Truth  
## Prediction yes no  
## yes 357 23  
## no 26 622

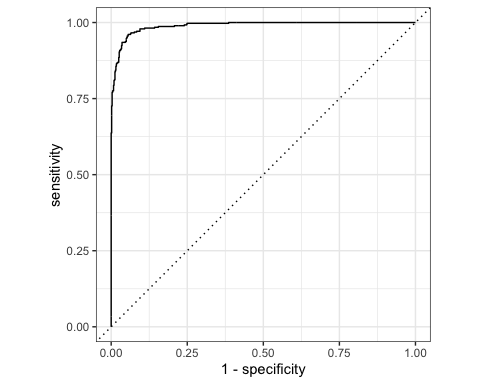
#### F1 score

f\_meas(test\_results,truth = loan\_default,estimate = .pred\_class)

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 f\_meas binary 0.936

#### ROC curve

autoplot(roc\_curve(test\_results,loan\_default,.pred\_yes))



# Area under the ROC curve  
roc\_auc(test\_results,loan\_default, .pred\_yes)

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 roc\_auc binary 0.989

#### Creating custom metric sets

my\_metrics <- metric\_set(yardstick::accuracy,f\_meas)  
my\_metrics(test\_results,truth=loan\_default,estimate=.pred\_class)

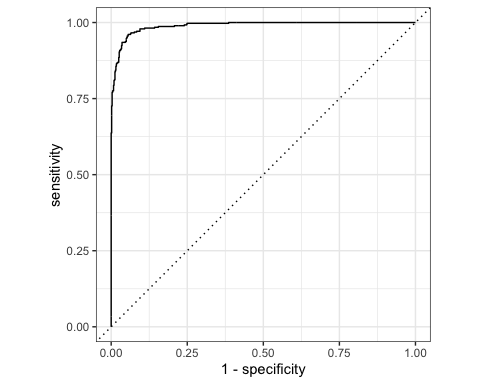
## # A tibble: 2 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 accuracy binary 0.952  
## 2 f\_meas binary 0.936

## Automate the process

# Automate the metrics process  
last\_fit\_model <- loan\_wf %>%  
 last\_fit(split = loan\_split)  
  
# Accuracy and area under ROC  
last\_fit\_model %>% collect\_metrics()

## # A tibble: 2 × 4  
## .metric .estimator .estimate .config   
## <chr> <chr> <dbl> <chr>   
## 1 accuracy binary 0.952 Preprocessor1\_Model1  
## 2 roc\_auc binary 0.989 Preprocessor1\_Model1

# Obtain data frames with predictions  
last\_fit\_results <- last\_fit\_model%>% collect\_predictions()  
  
  
last\_fit\_results %>%   
 roc\_curve(loan\_default,.pred\_yes) %>%  
 autoplot()

 ## Model 2

### Creating folds for cross validation on training dataset

# These will be used to tune hyperparameters  
set.seed(345)  
loan\_folds <- vfold\_cv(loan\_train, v=5)  
  
loan\_folds

## # 5-fold cross-validation   
## # A tibble: 5 × 2  
## splits id   
## <list> <chr>  
## 1 <split [2465/617]> Fold1  
## 2 <split [2465/617]> Fold2  
## 3 <split [2466/616]> Fold3  
## 4 <split [2466/616]> Fold4  
## 5 <split [2466/616]> Fold5

### Model specification

knn\_model <- nearest\_neighbor(neighbors = tune()) %>%  
 set\_engine('kknn') %>%  
 set\_mode('classification')  
  
knn\_model

## K-Nearest Neighbor Model Specification (classification)  
##   
## Main Arguments:  
## neighbors = tune()  
##   
## Computational engine: kknn

### Create Workflow

knn\_wf <- workflow() %>%  
 add\_model(knn\_model) %>%  
 add\_recipe(loan\_recipe)  
  
knn\_wf

## ══ Workflow ════════════════════════════════════════════════════════════════════  
## Preprocessor: Recipe  
## Model: nearest\_neighbor()  
##   
## ── Preprocessor ────────────────────────────────────────────────────────────────  
## 3 Recipe Steps  
##   
## • step\_YeoJohnson()  
## • step\_normalize()  
## • step\_dummy()  
##   
## ── Model ───────────────────────────────────────────────────────────────────────  
## K-Nearest Neighbor Model Specification (classification)  
##   
## Main Arguments:  
## neighbors = tune()  
##   
## Computational engine: kknn

### Hyperparameter Tuning - Create neighbor grid

k\_grid <- tibble(neighbors = c(10,20,30,50,75,100,125,150))  
k\_grid

## # A tibble: 8 × 1  
## neighbors  
## <dbl>  
## 1 10  
## 2 20  
## 3 30  
## 4 50  
## 5 75  
## 6 100  
## 7 125  
## 8 150

### Hyperparameter Tuning - Tune Workflow

set.seed(345)  
  
knn\_tunning <- knn\_wf%>%  
 tune\_grid(resamples = loan\_folds, grid = k\_grid)  
  
knn\_tunning

## # Tuning results  
## # 5-fold cross-validation   
## # A tibble: 5 × 4  
## splits id .metrics .notes   
## <list> <chr> <list> <list>   
## 1 <split [2465/617]> Fold1 <tibble [16 × 5]> <tibble [0 × 3]>  
## 2 <split [2465/617]> Fold2 <tibble [16 × 5]> <tibble [0 × 3]>  
## 3 <split [2466/616]> Fold3 <tibble [16 × 5]> <tibble [0 × 3]>  
## 4 <split [2466/616]> Fold4 <tibble [16 × 5]> <tibble [0 × 3]>  
## 5 <split [2466/616]> Fold5 <tibble [16 × 5]> <tibble [0 × 3]>

### Hyperparameter Tuning - Access Model Performance

knn\_tunning %>%  
 show\_best('roc\_auc')

## # A tibble: 5 × 7  
## neighbors .metric .estimator mean n std\_err .config   
## <dbl> <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 150 roc\_auc binary 0.907 5 0.00629 Preprocessor1\_Model8  
## 2 125 roc\_auc binary 0.905 5 0.00613 Preprocessor1\_Model7  
## 3 100 roc\_auc binary 0.904 5 0.00616 Preprocessor1\_Model6  
## 4 75 roc\_auc binary 0.902 5 0.00610 Preprocessor1\_Model5  
## 5 50 roc\_auc binary 0.897 5 0.00645 Preprocessor1\_Model4

# Select the best model based on roc\_auc  
best\_k <- knn\_tunning %>%  
 select\_best(metric = 'roc\_auc')  
best\_k

## # A tibble: 1 × 2  
## neighbors .config   
## <dbl> <chr>   
## 1 150 Preprocessor1\_Model8

### Hyperparameter Tuning - Finalize workflow with best performance model

final\_knn\_wf <- knn\_wf %>%  
 finalize\_workflow(best\_k)  
final\_knn\_wf

## ══ Workflow ════════════════════════════════════════════════════════════════════  
## Preprocessor: Recipe  
## Model: nearest\_neighbor()  
##   
## ── Preprocessor ────────────────────────────────────────────────────────────────  
## 3 Recipe Steps  
##   
## • step\_YeoJohnson()  
## • step\_normalize()  
## • step\_dummy()  
##   
## ── Model ───────────────────────────────────────────────────────────────────────  
## K-Nearest Neighbor Model Specification (classification)  
##   
## Main Arguments:  
## neighbors = 150  
##   
## Computational engine: kknn

### Fit the model

# Fit the model  
knn\_wf\_fit <- final\_knn\_wf %>%  
 fit(data=loan\_train)  
  
#Explore trained model  
knn\_fit <- knn\_wf\_fit %>%  
 extract\_fit\_parsnip()  
knn\_fit

## parsnip model object  
##   
##   
## Call:  
## kknn::train.kknn(formula = ..y ~ ., data = data, ks = min\_rows(150, data, 5))  
##   
## Type of response variable: nominal  
## Minimal misclassification: 0.1862427  
## Best kernel: optimal  
## Best k: 150

### Hyperparameter Tuning - Train, Evaluate and Collect metrics

# Train and Evaluate  
last\_fit\_knn <- final\_knn\_wf %>%  
 last\_fit(split = loan\_split)  
  
#Collect metrics  
last\_fit\_knn %>% collect\_metrics()

## # A tibble: 2 × 4  
## .metric .estimator .estimate .config   
## <chr> <chr> <dbl> <chr>   
## 1 accuracy binary 0.812 Preprocessor1\_Model1  
## 2 roc\_auc binary 0.892 Preprocessor1\_Model1

### Evaluate performance

pred\_categories <- predict(knn\_wf\_fit,new\_data = loan\_test)  
pred\_categories

## # A tibble: 1,028 × 1  
## .pred\_class  
## <fct>   
## 1 yes   
## 2 no   
## 3 yes   
## 4 no   
## 5 no   
## 6 no   
## 7 no   
## 8 yes   
## 9 no   
## 10 yes   
## # ℹ 1,018 more rows

#Predicting probabilities  
pred\_prob <- predict(knn\_wf\_fit,new\_data = loan\_test,type = 'prob')  
pred\_prob

## # A tibble: 1,028 × 2  
## .pred\_yes .pred\_no  
## <dbl> <dbl>  
## 1 0.624 0.376  
## 2 0.416 0.584  
## 3 0.835 0.165  
## 4 0.289 0.711  
## 5 0.237 0.763  
## 6 0.356 0.644  
## 7 0.373 0.627  
## 8 0.627 0.373  
## 9 0.272 0.728  
## 10 0.518 0.482  
## # ℹ 1,018 more rows

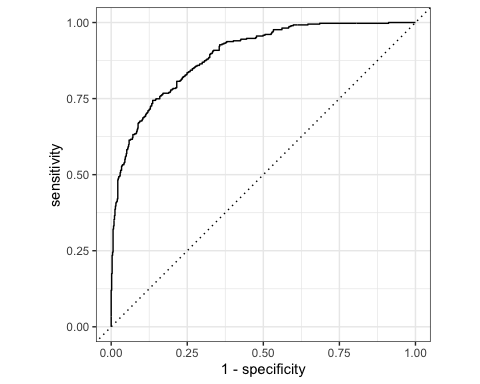
#Combine results from above with true response variable values in our test data set  
test\_results\_1 <- loan\_test %>%  
 select(loan\_default) %>%  
 bind\_cols(pred\_categories) %>%  
 bind\_cols(pred\_prob)  
test\_results\_1

## # A tibble: 1,028 × 4  
## loan\_default .pred\_class .pred\_yes .pred\_no  
## <fct> <fct> <dbl> <dbl>  
## 1 yes yes 0.624 0.376  
## 2 yes no 0.416 0.584  
## 3 yes yes 0.835 0.165  
## 4 no no 0.289 0.711  
## 5 no no 0.237 0.763  
## 6 no no 0.356 0.644  
## 7 yes no 0.373 0.627  
## 8 yes yes 0.627 0.373  
## 9 yes no 0.272 0.728  
## 10 no yes 0.518 0.482  
## # ℹ 1,018 more rows

### Exploring Performance Metrics

#### ROC Curve

# Collect predictions  
knn\_predictions <- last\_fit\_knn %>% collect\_predictions()  
  
knn\_predictions %>%  
 roc\_curve(loan\_default,.pred\_yes) %>%  
 autoplot()

 #### Confusion Matrix

conf\_mat(knn\_predictions,loan\_default,.pred\_class)

## Truth  
## Prediction yes no  
## yes 223 33  
## no 160 612

#### F1 score

f\_meas(test\_results\_1,truth = loan\_default,estimate = .pred\_class)

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 f\_meas binary 0.698

### Creating custom metric set

my\_metrics <- metric\_set(yardstick::accuracy,f\_meas)  
my\_metrics(test\_results\_1,truth=loan\_default,estimate=.pred\_class)

## # A tibble: 2 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 accuracy binary 0.812  
## 2 f\_meas binary 0.698

## Model 3

skim(loans\_df)

Data summary

|  |  |
| --- | --- |
| Name | loans\_df |
| Number of rows | 4110 |
| Number of columns | 16 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 8 |
| numeric | 8 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| --- | --- | --- | --- | --- | --- |
| loan\_default | 0 | 1 | FALSE | 2 | no: 2580, yes: 1530 |
| loan\_purpose | 0 | 1 | FALSE | 5 | deb: 1218, cre: 879, sma: 853, med: 635 |
| application\_type | 0 | 1 | FALSE | 2 | ind: 3494, joi: 616 |
| term | 0 | 1 | FALSE | 2 | thr: 2588, fiv: 1522 |
| homeownership | 0 | 1 | FALSE | 3 | mor: 1937, ren: 1666, own: 507 |
| missed\_payment\_2\_yr | 0 | 1 | FALSE | 2 | no: 3640, yes: 470 |
| history\_bankruptcy | 0 | 1 | FALSE | 2 | no: 3624, yes: 486 |
| history\_tax\_liens | 0 | 1 | FALSE | 2 | no: 4050, yes: 60 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| loan\_amount | 0 | 1 | 16692.79 | 10038.89 | 1000.00 | 9600.00 | 15000.00 | 24000.00 | 40000.00 | ▆▇▅▃▂ |
| installment | 0 | 1 | 489.42 | 289.50 | 31.04 | 274.82 | 421.97 | 663.98 | 1566.59 | ▇▇▅▂▁ |
| interest\_rate | 0 | 1 | 11.38 | 3.92 | 4.72 | 8.22 | 11.25 | 13.75 | 20.00 | ▆▆▇▃▃ |
| annual\_income | 0 | 1 | 73015.01 | 37203.11 | 3000.00 | 45000.00 | 65000.00 | 92000.00 | 200000.00 | ▃▇▃▁▁ |
| current\_job\_years | 0 | 1 | 5.80 | 3.69 | 0.00 | 2.00 | 5.00 | 10.00 | 10.00 | ▆▃▂▂▇ |
| debt\_to\_income | 0 | 1 | 20.04 | 14.23 | 0.00 | 11.85 | 18.59 | 26.13 | 437.61 | ▇▁▁▁▁ |
| total\_credit\_lines | 0 | 1 | 22.47 | 12.03 | 2.00 | 14.00 | 20.00 | 29.00 | 87.00 | ▇▇▂▁▁ |
| years\_credit\_history | 0 | 1 | 15.76 | 7.22 | 3.00 | 11.00 | 14.00 | 19.00 | 51.00 | ▆▇▂▁▁ |

### Model Specification

tree\_model <- decision\_tree(cost\_complexity = tune(),  
 tree\_depth = tune(), min\_n = tune()) %>%  
 set\_engine('rpart') %>%  
 set\_mode('classification')  
tree\_model

## Decision Tree Model Specification (classification)  
##   
## Main Arguments:  
## cost\_complexity = tune()  
## tree\_depth = tune()  
## min\_n = tune()  
##   
## Computational engine: rpart

### Execute a workflow

tree\_wf <- workflow() %>%  
 add\_model(tree\_model) %>%  
 add\_recipe(loan\_recipe)

### Hyperparameter Tuning - Create grid

tree\_grid <- grid\_regular(cost\_complexity(),  
 tree\_depth(),  
 min\_n(),  
 levels = 2)  
tree\_grid

## # A tibble: 8 × 3  
## cost\_complexity tree\_depth min\_n  
## <dbl> <int> <int>  
## 1 0.0000000001 1 2  
## 2 0.1 1 2  
## 3 0.0000000001 15 2  
## 4 0.1 15 2  
## 5 0.0000000001 1 40  
## 6 0.1 1 40  
## 7 0.0000000001 15 40  
## 8 0.1 15 40

### Tuning hyperparameters

set.seed(345)  
tree\_tuning <- tree\_wf %>%  
 tune\_grid(resamples = loan\_folds, grid = tree\_grid)

### Top 5 models

tree\_tuning %>% show\_best('roc\_auc')

## # A tibble: 5 × 9  
## cost\_complexity tree\_depth min\_n .metric .estimator mean n std\_err  
## <dbl> <int> <int> <chr> <chr> <dbl> <int> <dbl>  
## 1 0.0000000001 15 40 roc\_auc binary 0.963 5 0.00623  
## 2 0.0000000001 15 2 roc\_auc binary 0.916 5 0.0125   
## 3 0.0000000001 1 2 roc\_auc binary 0.804 5 0.00598  
## 4 0.1 1 2 roc\_auc binary 0.804 5 0.00598  
## 5 0.1 15 2 roc\_auc binary 0.804 5 0.00598  
## # ℹ 1 more variable: .config <chr>

# Select best model based on roc\_auc  
best\_tree <- tree\_tuning %>%  
 select\_best(metric = 'roc\_auc')  
  
best\_tree

## # A tibble: 1 × 4  
## cost\_complexity tree\_depth min\_n .config   
## <dbl> <int> <int> <chr>   
## 1 0.0000000001 15 40 Preprocessor1\_Model7

### Finalize workflow

final\_tree\_wf <- tree\_wf %>%  
 finalize\_workflow(best\_tree)  
final\_tree\_wf

## ══ Workflow ════════════════════════════════════════════════════════════════════  
## Preprocessor: Recipe  
## Model: decision\_tree()  
##   
## ── Preprocessor ────────────────────────────────────────────────────────────────  
## 3 Recipe Steps  
##   
## • step\_YeoJohnson()  
## • step\_normalize()  
## • step\_dummy()  
##   
## ── Model ───────────────────────────────────────────────────────────────────────  
## Decision Tree Model Specification (classification)  
##   
## Main Arguments:  
## cost\_complexity = 1e-10  
## tree\_depth = 15  
## min\_n = 40  
##   
## Computational engine: rpart

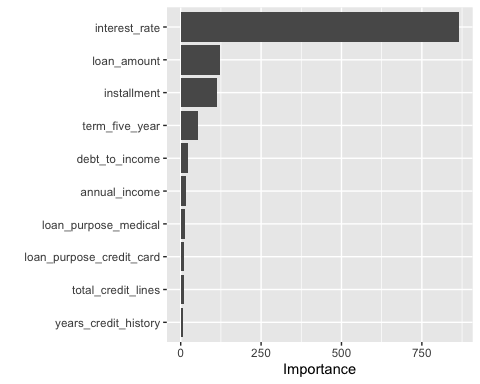
### Visualize results

# Fit the model  
tree\_wf\_fit <- final\_tree\_wf %>%  
 fit(data = loan\_train)

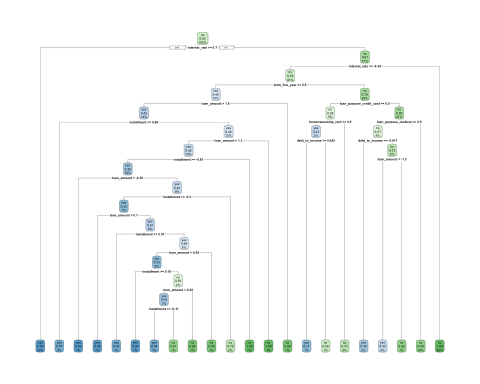
tree\_fit <- tree\_wf\_fit %>%  
 extract\_fit\_parsnip()

### Variable importance

vip(tree\_fit)

 ### Decision tree plot

rpart.plot(tree\_fit$fit, roundint = FALSE)

 ### Train and Evaluate

tree\_last\_fit <- final\_tree\_wf %>%  
 last\_fit(loan\_split)

### Evaluate performance

pred\_categories <- predict(tree\_wf\_fit,new\_data = loan\_test)  
pred\_categories

## # A tibble: 1,028 × 1  
## .pred\_class  
## <fct>   
## 1 yes   
## 2 yes   
## 3 yes   
## 4 yes   
## 5 no   
## 6 no   
## 7 yes   
## 8 yes   
## 9 yes   
## 10 yes   
## # ℹ 1,018 more rows

#Predicting probabilities  
pred\_prob <- predict(tree\_wf\_fit,new\_data = loan\_test,type = 'prob')  
pred\_prob

## # A tibble: 1,028 × 2  
## .pred\_yes .pred\_no  
## <dbl> <dbl>  
## 1 1 0   
## 2 0.571 0.429   
## 3 1 0   
## 4 0.966 0.0345  
## 5 0.108 0.892   
## 6 0 1   
## 7 0.604 0.396   
## 8 1 0   
## 9 1 0   
## 10 0.730 0.270   
## # ℹ 1,018 more rows

#Combine results from above with true response variable values in our test data set  
test\_results\_2<- loan\_test %>%  
 select(loan\_default) %>%  
 bind\_cols(pred\_categories) %>%  
 bind\_cols(pred\_prob)  
test\_results\_2

## # A tibble: 1,028 × 4  
## loan\_default .pred\_class .pred\_yes .pred\_no  
## <fct> <fct> <dbl> <dbl>  
## 1 yes yes 1 0   
## 2 yes yes 0.571 0.429   
## 3 yes yes 1 0   
## 4 no yes 0.966 0.0345  
## 5 no no 0.108 0.892   
## 6 no no 0 1   
## 7 yes yes 0.604 0.396   
## 8 yes yes 1 0   
## 9 yes yes 1 0   
## 10 no yes 0.730 0.270   
## # ℹ 1,018 more rows

### Accuracy and Area under the ROC curve

tree\_last\_fit %>% collect\_metrics()

## # A tibble: 2 × 4  
## .metric .estimator .estimate .config   
## <chr> <chr> <dbl> <chr>   
## 1 accuracy binary 0.915 Preprocessor1\_Model1  
## 2 roc\_auc binary 0.965 Preprocessor1\_Model1

### Collect predictions

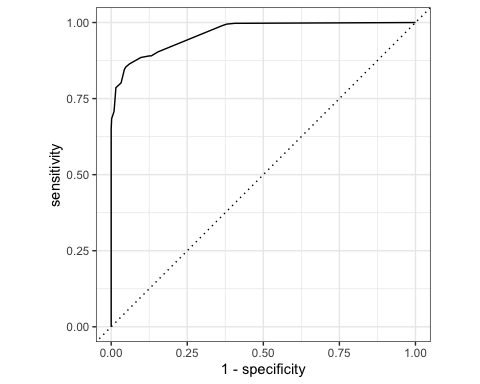
tree\_prediction <- tree\_last\_fit %>% collect\_predictions()  
tree\_prediction

## # A tibble: 1,028 × 7  
## id .pred\_yes .pred\_no .row .pred\_class loan\_default .config   
## <chr> <dbl> <dbl> <int> <fct> <fct> <chr>   
## 1 train/test split 1 0 1 yes yes Preproces…  
## 2 train/test split 0.571 0.429 4 yes yes Preproces…  
## 3 train/test split 1 0 6 yes yes Preproces…  
## 4 train/test split 0.966 0.0345 8 yes no Preproces…  
## 5 train/test split 0.108 0.892 10 no no Preproces…  
## 6 train/test split 0 1 17 no no Preproces…  
## 7 train/test split 0.604 0.396 21 yes yes Preproces…  
## 8 train/test split 1 0 24 yes yes Preproces…  
## 9 train/test split 1 0 26 yes yes Preproces…  
## 10 train/test split 0.730 0.270 28 yes no Preproces…  
## # ℹ 1,018 more rows

### Exploring performance metrics

#### ROC curve

tree\_prediction %>%  
 roc\_curve(loan\_default, .pred\_yes) %>%  
 autoplot()

 #### Confusion matrix

conf\_mat(tree\_prediction, loan\_default, .pred\_class)

## Truth  
## Prediction yes no  
## yes 327 31  
## no 56 614

#### F1 score

f\_meas(test\_results\_2,truth = loan\_default,estimate = .pred\_class)

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 f\_meas binary 0.883

# Summary of Results [50 Points]

**1. Introduction**

This company wants to determine the factors causing default on loans and build a model to predict the likelihood of an applicant defaulting on loans. Since lending money is the major business operation of the company, loan defaulting can negatively impact the company’s net income and profit. That is why they want to minimize the risk by declining those who are likely not to pay back, but at the same time, they don’t want to reject customers who have the potential to pay back both the capital and the interest on time. Our job as data analysts are to build and select the best predicting model so that the company can make better decisions on when to decline an application and when to accept one.

**2. Highlights and key findings from Exploratory Data Analysis**

The dataset consists of 16 variables and 4110 observations. Exploratory data analysis shows that people who default on loans have lower annual income, have higher interest rates, and pay higher each month compared to those who don’t default on loans. Interestingly, homeowners are the group that has the lowest loan default rate, and renters make up the majority of loan defaulting. The brief analysis gives us an idea of the factors affecting the ability to pay back loans. To precisely use the combination of these factors to decide whether to lend someone money, we need to build a model that yields the most optimal decision whenever someone submits an application for a loan.

**3. Your “best” classification model and an analysis of its performance**

Based on the accuracy, F1 score, and area under the curve (AUC), the logistic regression model is the best-predicting model for this business scenario. Accuracy is calculated based on the confusion matrix. The formula for accuracy is: (TP + TN)/(TP + FP +FN + FP) TP: True Positive TN: True Negative FP: False Positive FN: False Negative The higher the accuracy, the better the model. Among the three models, logistic regression has the highest accuracy score, 0.952 or 95.2%. The KNN model has the lowest accuracy score of 0.812 or 81.2%. F1 score is an error metric ranging from 0 to 1, with 0 representing the worst and 1 being the best. Again, among the three models, the logistic model has the highest F1 score of 0.935, followed by the decision tree model with 0.88, and KNN has the lowest score of 0.7. The final metric used to determine the best model is the area under the ROC curve (AUC). This metric tells us how well a model performs case separation (classifying loan\_default and no\_loan\_default). The higher the number, the better the model at classifying the two categories. Among the three models, the logistic model has the highest AUC of 0.989, followed by the decision tree with 0.965, and the KNN model with 0.892. In short, based on these three metrics - accuracy, F1 score, and AUC, we conclude that the logistic regression model is the best for predicting the potentiality of loan default based on the top four factors - loan terms, monthly payment, loan amount, and interest rate.

**4. Your recommendations to the company on how to reduce loan default rates**

The top four factors affecting loan default determined by the logistic regression model are - loan term, monthly payment, loan amount, and interest rate. All the four elements determined above strongly interrelate. When people apply for a loan, their interest rate and monthly payment are determined by multiple factors such as annual income, history of late payment or bankruptcy, credit score, etc. For some applicants, interest rates and monthly payments are high due to bad credit history. Bad credit history is a red flag for lenders because the borrowers can delay or default on loans like they did in the past. In addition, when people take out a large sum of money, they tend to have longer-term loans. However, the longer the term, the more likely they default on the loan.

Recommendations:

* Check the credit history of the applicant within the last 5 years. Any loan default, late, or missed payment need to be investigated. If an applicant miss payment or pay late more than two times within a year, he should not be approved for the loan.
* Limit the amount of loan. The model indicates that people who borrow more tend to default on the loan.

The primary source of income for the bank is the difference between the money paid from loans and the money the bank pays out. Hence, if there are many loan defaults, or the difference between the money received and the money paid is negative, the bank will lose revenue and earn no profit. This is why the bank wants to have the minimum loan defaults possible.

**5. Conclusion**

The logistic regression model is the best model for this business situation with an accuracy of 95% and classification capability almost perfect, 98.9 out of 100. The model also identifies the top 4 factors explaining why customers default on loans. In conclusion, the logistic regression model will help the bank classify applicants who are likely to default on loans and those who won’t