# Analytics Project

**Your Name**: IWAN **Your G Number**: G01236399

```r
# Suppress dplyr summarise grouping warning messages
options(dplyr.summarise.inform = FALSE)

## Add R libraries here
library(tidyverse)
library(tidymodels)
library(vip)
library(rpart.plot)

# Load data
loans_df <- read_rds(url('https://gmubusinessanalytics.netlify.app/data/loan_data.rds'))
```

## Data Analysis [50 Points]

In this section, you must think of at least 5 relevant questions that explore the relationship between `loan_default` and the other variables in the `loan_df` data set. The goal of your analysis should be discovering which variables drive the differences between customers who do and do not default on their loans.

You must answer each question and provide supporting data summaries with either a summary data frame (using `dplyr/tidyr`) or a plot (using `ggplot`) or both.

In total, you must have a minimum of 3 plots (created with `ggplot`) and 3 summary data frames (created with `dplyr`) for the exploratory data analysis section. Among the plots you produce, you must have at least 3 different types (ex. box plot, bar chart, histogram, scatter plot, etc...)

See the example question below.

**Note**: To add an R code chunk to any section of your project, you can use the keyboard shortcut `Ctrl` + `Alt` + `i` or the `insert` button at the top of your R project template notebook file.

### Sample Question

#### Are there differences in loan default rates by loan purpose?

**Answer**: Yes, the data indicates that credit card and medical loans have significantly larger default rates than any other type of loan. In fact, both of these loan types have default rates at more than 50%. This is nearly two times the average default rate for all other loan types.

#### Summary Table

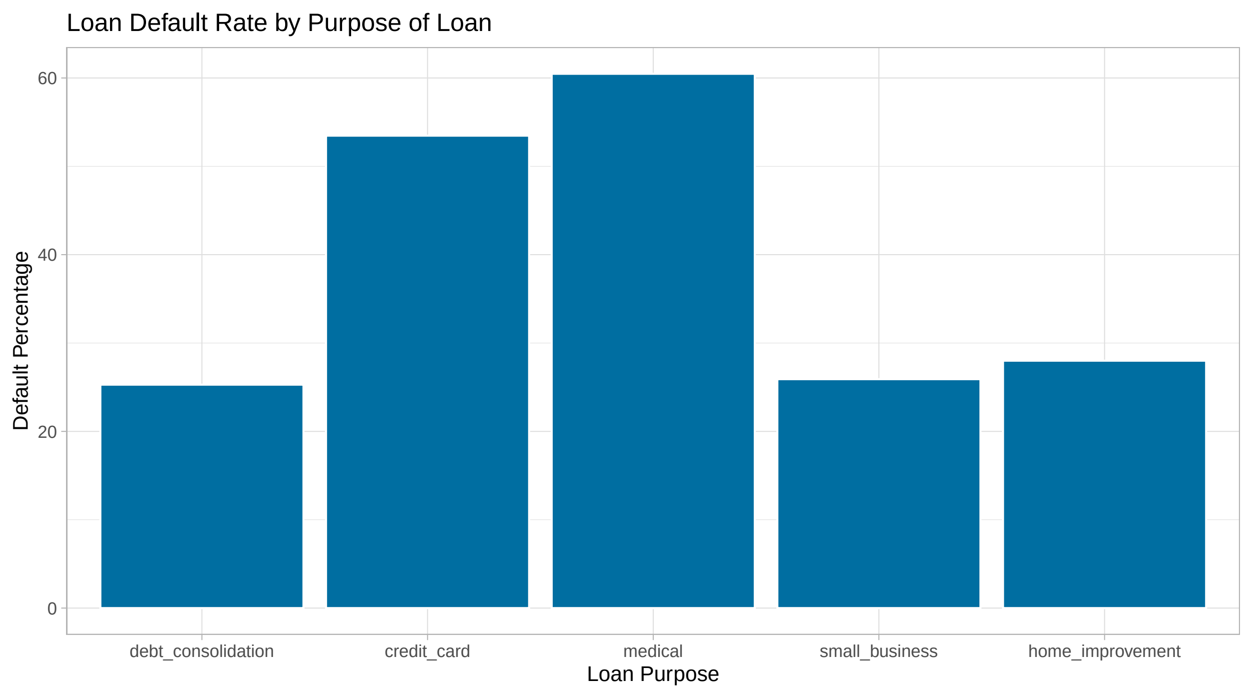
```r
loans_df %>%
  group_by(loan_purpose) %>%
  summarise(n_customers = n(),
            customers_default = sum(loan_default == 'yes'),
            default_percent = 100 * mean(loan_default == 'yes'))
```

```
## # A tibble: 5 x 4
##   loan_purpose       n_customers customers_default default_percent
##   <fct>                    <int>             <int>           <dbl>
## 1 debt_consolidation        1218               308            25.3
## 2 credit_card                879               470            53.5
## 3 medical                    635               384            60.5
## 4 small_business             853               221            25.9
## 5 home_improvement           525               147            28.
```

### Data Visulatization

```
default_rates <- loans_df %>%
                 group_by(loan_purpose) %>%
                 summarise(n_customers = n(),
                 customers_default = sum(loan_default == 'yes'),
                 default_percent = 100 * mean(loan_default == 'yes'))


ggplot(data = default_rates,
       mapping = aes(x = loan_purpose,
                     y = default_percent)) +
    geom_bar(stat = 'identity',
             fill = '#006EA1',
             color = 'white') +
    labs(title = 'Loan Default Rate by Purpose of Loan',
         x = 'Loan Purpose',
         y = 'Default Percentage') +
    theme_light()
```



## Question 1

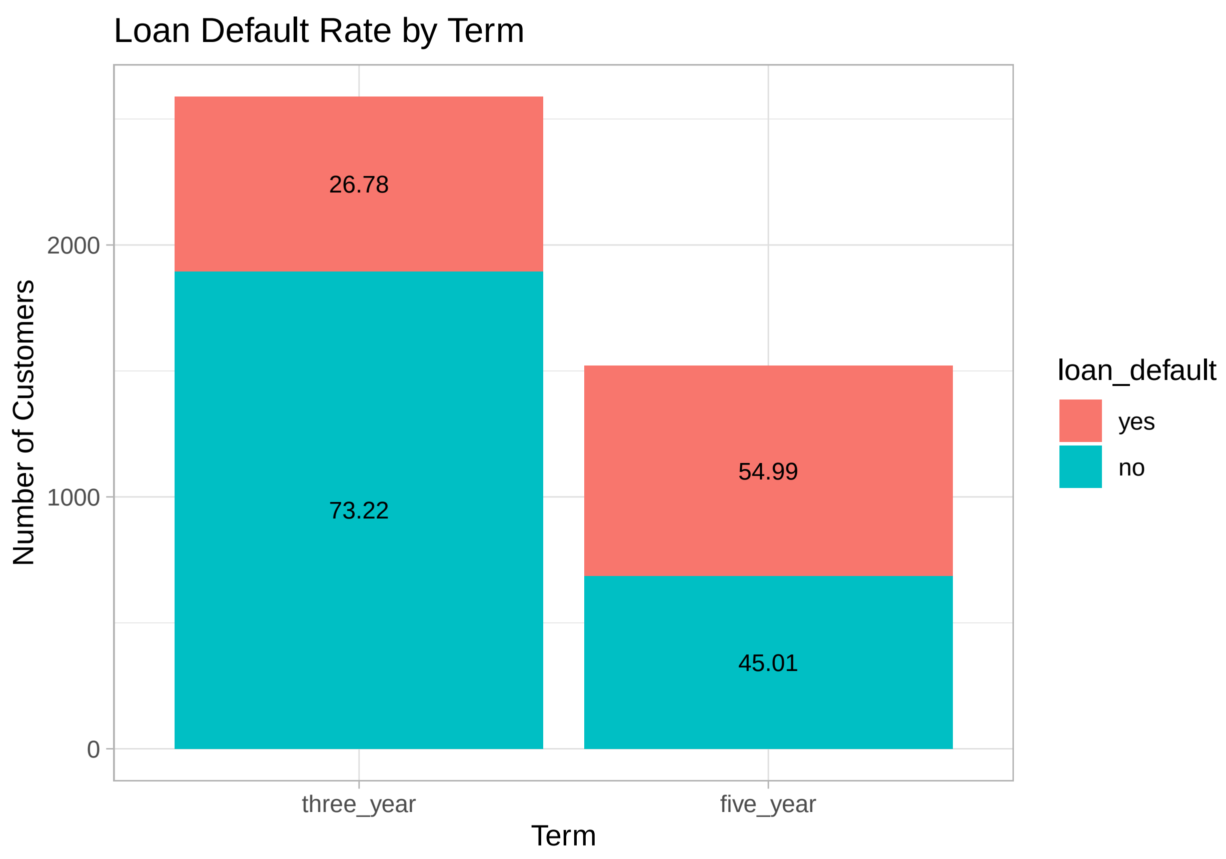
**Question**: Are there differences in loan default rates by term?

**Answer**: Yes, there are difference in it. It precisely shown by the plot that the percentage of loan default in five years term has the higher rate compared to the three years term. However, the non default customers is likely to happen to happen in the three years term instead of in the five years term.

```
default_rates2_1 <- loans_df %>%
  group_by(term, loan_default) %>%
  summarise(n_customers = n()) %>%
  mutate(percentage = (n_customers/sum(n_customers)) * 100)
default_rates2_1
```

```
## # A tibble: 4 x 4
## # Groups:   term [2]
##   term       loan_default n_customers percentage
##   <fct>      <fct>              <int>      <dbl>
## 1 three_year yes                  693       26.8
## 2 three_year no                  1895       73.2
## 3 five_year  yes                  837       55.0
## 4 five_year  no                   685       45.0
```

```
ggplot(data = default_rates2_1,
       mapping = aes(x = term,
                     y = n_customers,
                     fill = loan_default)) +
  geom_bar(stat = 'identity') +
  geom_text(aes(label = round(percentage,
                              digits = 2)),
            size = 3,
            position = position_stack(vjust = 0.5)) +
  labs(title = 'Loan Default Rate by Term',
       x = 'Term',
       y = 'Number of Customers') +
  theme_light()
```



## Question 2

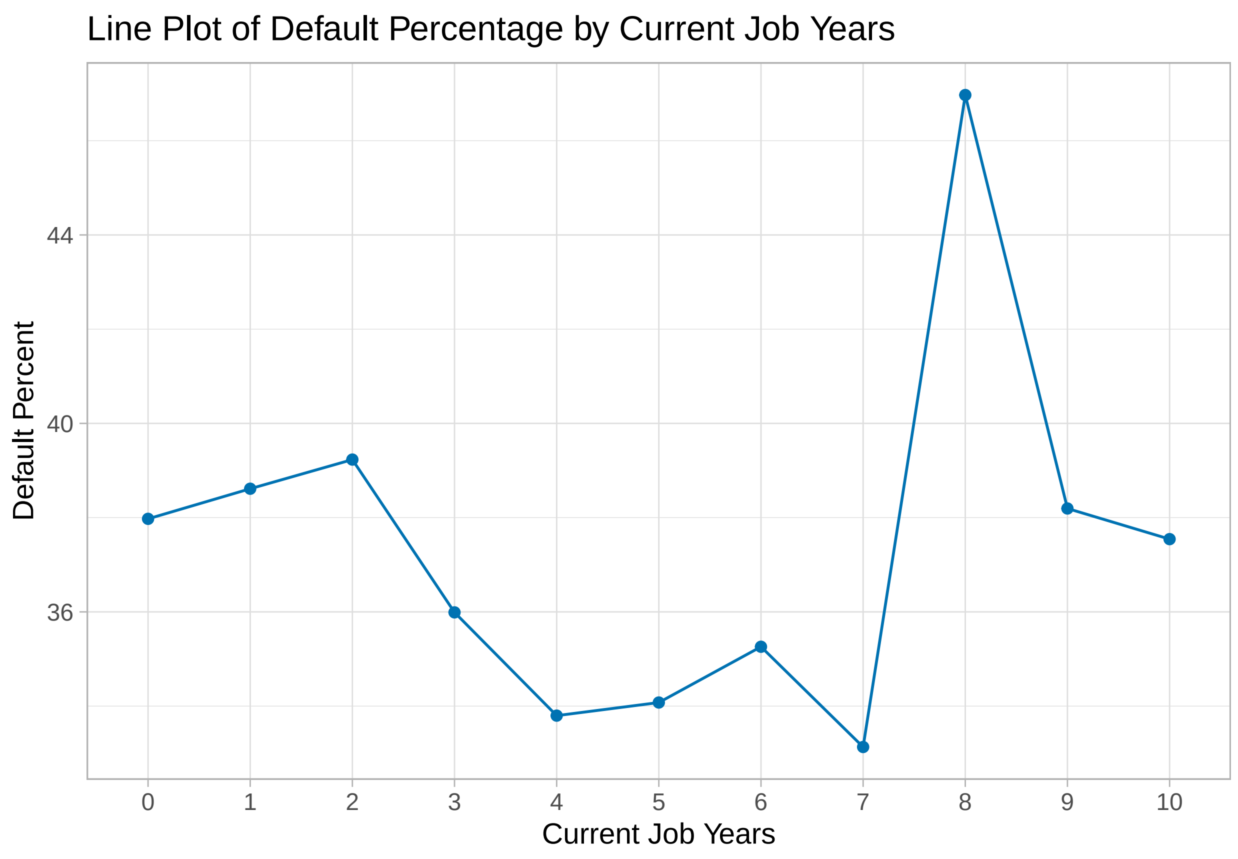
**Question**: Are there differences in loan default rates by current_job_years?

**Answer**: Yes, there are differences. It can clearly be exaplained by the line plot that the customers, which have the current job years in the middle year from 4 to 7 years, are likely not to default in the loan payment. However, there is a significant number of loan default customer spike for the customers in 8 years of work duration. It automatically explains that the most number of customers who likely to default in loan payment are customers which have the 8 years duration in current job years.

```
default_rates3 <- loans_df %>%
  group_by(current_job_years = as.factor(loans_df$current_job_years)) %>%
  summarise(n_customers = n(),
            customers_default = sum(loan_default == 'yes')) %>%
  mutate(default_percent = (customers_default/n_customers) * 100)
default_rates3
```

```
## # A tibble: 11 x 4
##    current_job_years n_customers customers_default default_percent
##    <fct>                   <int>             <int>           <dbl>
##  1 0                         316               120            38.0
##  2 1                         303               117            38.6
##  3 2                         469               184            39.2
##  4 3                         389               140            36.0
##  5 4                         287                97            33.8
##  6 5                         314               107            34.1
##   7 6                              173           61            35.3
##   8 7                              166           55            33.1
##   9 8                              132           62            47.0
##  10 9                              144           55            38.2
##  11 10                            1417          532            37.5
```

```
ggplot(data = default_rates3,
       mapping = aes(x = current_job_years,
                     y = default_percent,
                     group = 1)) +
  geom_line(color = "#0072B2") +
  geom_point(color = "#0072B2") +
  labs(title = "Line Plot of Default Percentage by Current Job Years",
       x = "Current Job Years",
       y = "Default Percent") +
  theme_light()
```

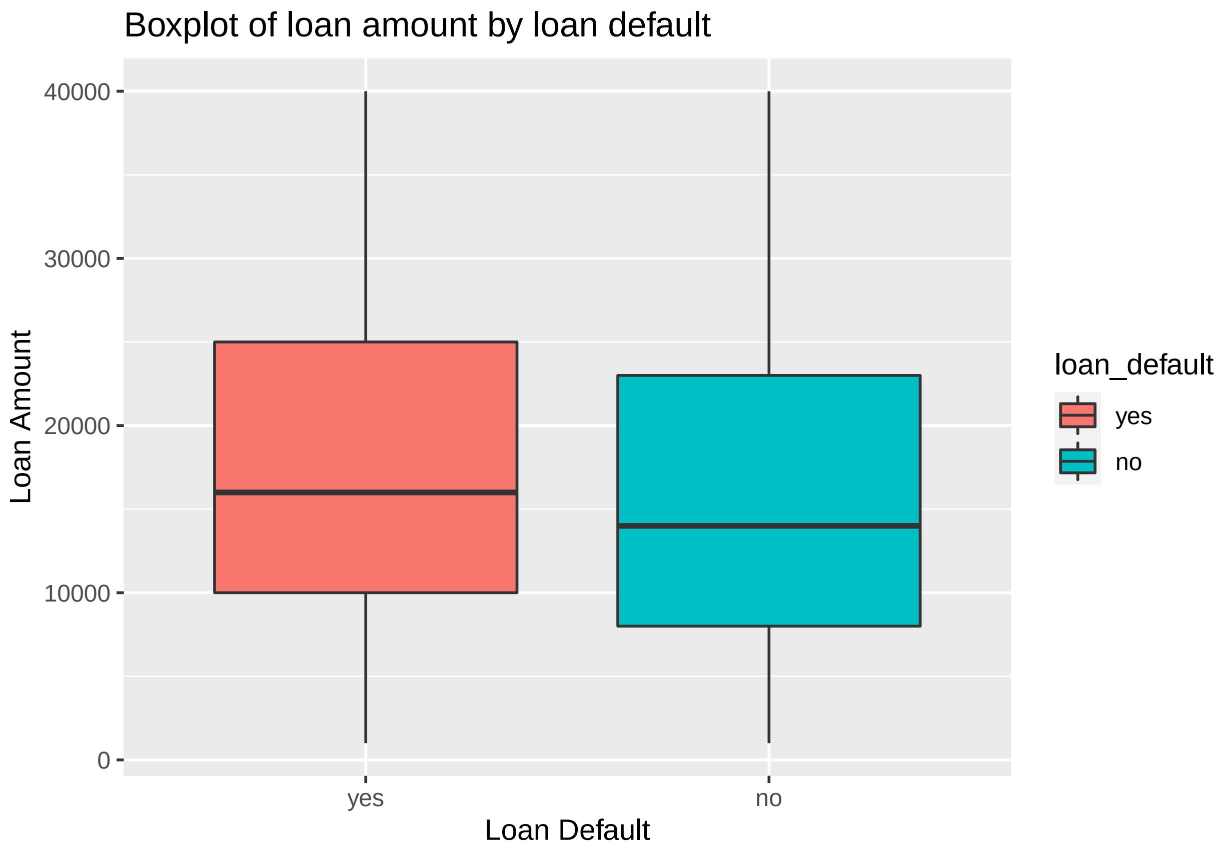


## Question 3

**Question**: Do the default customers have the higher median of Loan Amount?

**Answer**: Yes, they do. The boxplot shows that the loan default has the higher median than the non default one. It can be conluded that the customers which likely to have the loan default status have the higher exposure of loan amount instead of the non default customer.

```
ggplot(data = loans_df,
       mapping = aes(x = loan_default,
                    y = loan_amount,
                    fill = loan_default)) +
  geom_boxplot() +
  labs(title = "Boxplot of loan amount by loan default",
       x = "Loan Default",
       y = "Loan Amount")
```



## Question 4

**Question**: Are there dfferences in loan default based on the customers installment range?

**Answer**: Yes, there are. According to the installment range, it shows something interesting. The the loan default customers are likely to spike as the installment amount are getting higher. However, the non default percent of the customers are getting lower as the installment range are getting lower. Still, the average percentage of the loan default is lower than the non default customers.

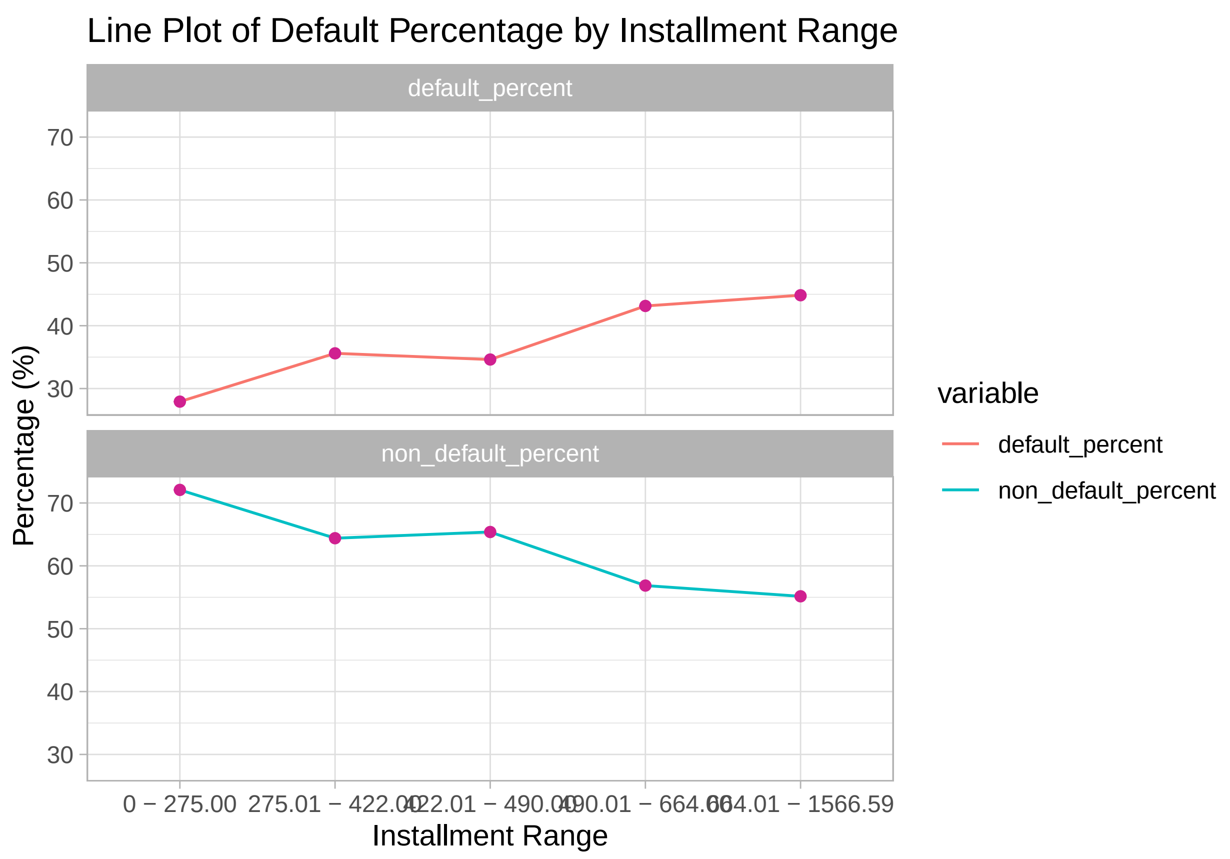
```
installment_differences <- loans_df %>%
  mutate(installment_range = cut(installment,
                               breaks = c(0,
                                          275.00,
                                          422.00,
                                          490.00,
                                          664.00,
                                          1566.59),
                               labels = c("0 - 275.00",
                                          "275.01 - 422.00",
```

```r
                                            "422.01 - 490.00",
                                            "490.01 - 664.00",
                                            "664.01 - 1566.59"))) %>%
  group_by(installment_range) %>%
  summarise(n_customer = n(),
            customer_default = sum(loan_default == 'yes'),
            customer_non_default = sum(loan_default == 'no'))%>%
  mutate(default_percent = (customer_default/n_customer) * 100,
         non_default_percent = (customer_non_default/n_customer) * 100)
installment_differences
```

```
## # A tibble: 5 x 6
##   installment_ran~ n_customer customer_default customer_non_de~ default_percent
##   <fct>                 <int>            <int>            <int>           <dbl>
## 1 0 - 275.00             1028              287              741            27.9
## 2 275.01 - 422.00        1028              366              662            35.6
## 3 422.01 - 490.00         312              108              204            34.6
## 4 490.01 - 664.00         714              308              406            43.1
## 5 664.01 - 1566.59       1028              461              567            44.8
## # ... with 1 more variable: non_default_percent <dbl>
```

```r
#preparation for the plot
df_installment_differences <- installment_differences %>%
  select(installment_range,
         default_percent,
         non_default_percent) %>%
  gather(key = "variable",
         value = "value",
         -installment_range)

#plot the loan default based on the range installment
ggplot(df_installment_differences,
       aes(x = installment_range,
           y = value,
           group = variable,
           color = variable)) +
  geom_line() +
  geom_point(color="violetred") +
  facet_wrap(~ variable, ncol = 1) +
  labs(title = "Line Plot of Default Percentage by Installment Range",
       x = "Installment Range",
       y = "Percentage (%)") +
  theme_light()
```



## Question 5

**Question**: Are there differences in loan default rates by homeownership?

**Answer**: Yes, there are. According to summary, the customers, whose home ownership status is "rent", have the higher percentage of the "mortgage" and the "own" status. Therefore, it can be concluded that the customers, who are renting house to live, have tendency to have the loan default status.

```
loans_df %>%
  group_by(homeownership) %>%
  summarise(n_customers = n(),
            customers_default = sum(loan_default == 'yes')) %>%
  mutate(default_percent = (customers_default/n_customers) * 100)
```

```
## # A tibble: 3 x 4
##   homeownership n_customers customers_default default_percent
##   <fct>               <int>             <int>           <dbl>
## 1 mortgage             1937               628            32.4
## 2 rent                 1666               713            42.8
## 3 own                   507               189            37.3
```

## Predictive Modeling [75 Points]

In this section of the project, you will fit **two classification algorithms** to predict the response variable,`loan_default`. You should use all of the other variables in the `loans_df` data as predictor variables for each model.

You must follow the machine learning steps below.

The data splitting and feature engineering steps should only be done once so that your models are using the same data and feature engineering steps for training.

- Split the `loans_df` data into a training and test set (remember to set your seed)
- Specify a feature engineering pipeline with the `recipes` package
  - You can include steps such as skewness transformation, dummy variable encoding or any other steps you find appropriate
- Specify a `parsnip` model object
  - You may choose from the following classification algorithms:
    * Logistic Regression
    * LDA
    * QDA
    * KNN
    * Decision Tree
    * Random Forest
- Package your recipe and model into a workflow
- Fit your workflow to the training data
  - If your model has hyperparameters:
    * Split the training data into 5 folds for 5-fold cross validation using `vfold_cv` (remember to set your seed)
    * Perform hyperparamter tuning with grid search using the `grid_regular()` function
    * Hyperparameter tuning can take a significant amount of computing time. To minimize this, use a maximum of 3 levels within your `grid_regular()` function
    * Select the best model with `select_best()` and finalize your workflow
- Evaluate model performance on the test set by plotting an ROC curve using `autoplot()` and calculating the area under the ROC curve on your test data

## Split the 'loans_df' data into a training and test set (remember to set your seed)

```r
#1. Split the `loans_df` data into a training and test set (remember to set your seed)
#--------------------------------------------------------------------------------------
set.seed(4500)
loans_split <- initial_split(loans_df, prop = 0.75,
                             strata = loan_default)

loans_training <- loans_split %>% training()
loans_test <- loans_split %>% testing()
```

## Specify a feature engineering pipeline

```r
#2. Specify a feature engineering pipeline with the `recipes` package
#--------------------------------------------------------------------------------------
loans_recipe <- recipe(loan_default ~ .,
                       data = loans_training) %>%
  step_YeoJohnson(all_numeric(),
                  -all_outcomes()) %>%
  step_normalize(all_numeric(),
                 -all_outcomes()) %>%
  step_dummy(all_nominal(),
             -all_outcomes())

loans_recipe %>%
  prep() %>%
  bake(new_data = loans_training)
```

```
## # A tibble: 3,083 x 20
##    loan_amount installment interest_rate annual_income current_job_yea~
##          <dbl>       <dbl>         <dbl>         <dbl>            <dbl>
##  1      1.56        1.38          1.40          0.941            -1.00
##  2     -0.550      -0.741         0.0985       -0.293             1.10
##  3     -0.216      -0.167         1.95          0.192             1.10
##  4      1.56        1.38          1.60          1.99             -1.74
##  5      0.958       0.602         0.215         1.33             -0.126
##  6     -1.23       -1.22         -0.860         0.108            -0.401
##  7      0.0173      0.0659        0.0280       -0.860             1.10
##  8      1.62        1.31          0.752        -1.75              1.10
##  9     -0.899      -0.675         1.80         -1.32             -0.126
## 10      1.62        0.993        -1.09          2.24              1.10
## # ... with 3,073 more rows, and 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>,
## #   missed_payment_2_yr_no <dbl>, history_bankruptcy_no <dbl>,
## #   history_tax_liens_no <dbl>
```

## Model 1

```
#3. Specify a `parsnip` model object
#-------------------------------------------------------------------------------------
# Model Specification
#-------------------------------------------------------------------------------------
# Logistic Regression
#-------------------------------------------------------------------------------------
logistic_model <- logistic_reg() %>%
  set_engine('glm') %>%
  set_mode('classification')
logistic_model
```

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

```
#4. Package your recipe and model into a workflow
#-------------------------------------------------------------------------------------
# Create a Workflow for logistic regression
#-------------------------------------------------------------------------------------
loans_wf <- workflow() %>%
  add_model(logistic_model) %>%
  add_recipe(loans_recipe)
loans_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
```

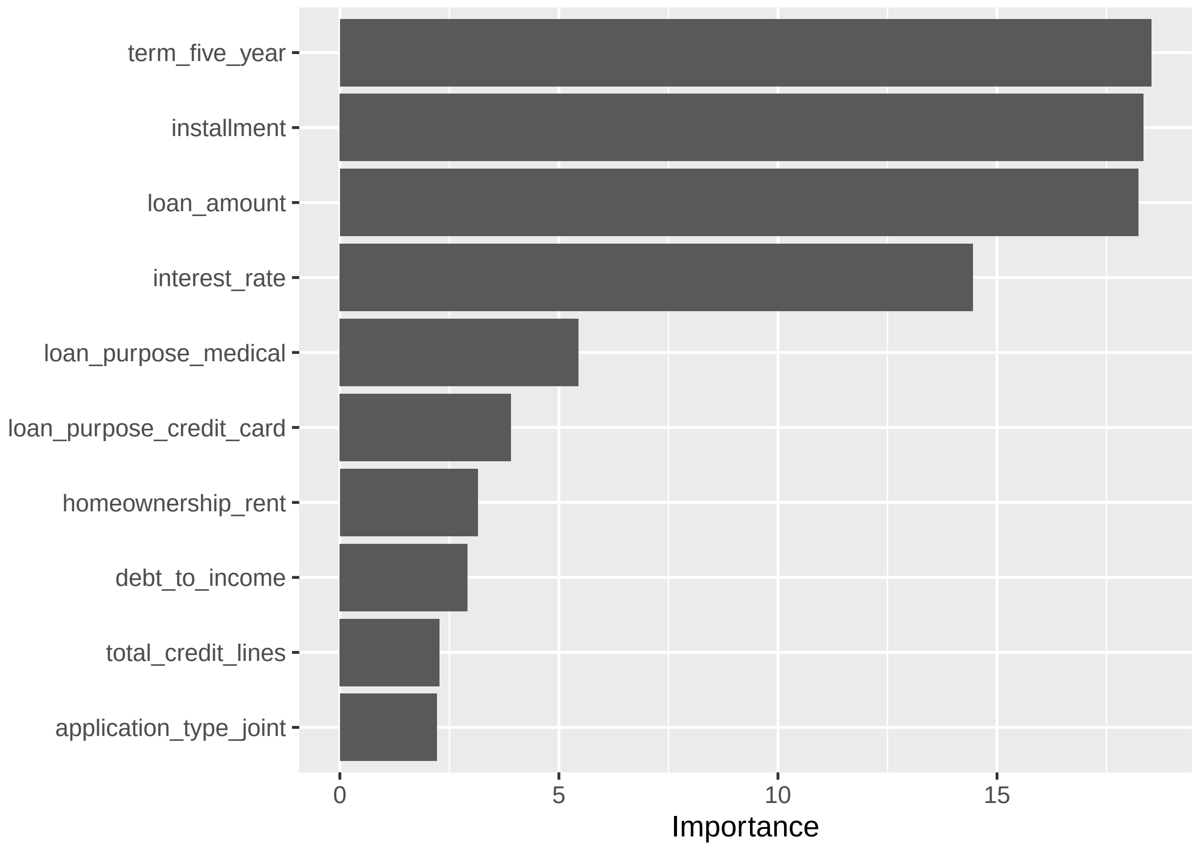
```
#5. Fit your workflow to the training data
#-----------------------------------------------------------------------------------
# Fit the Model to the logistic regression model
#-----------------------------------------------------------------------------------
# Fit the workflow to the training data
loans_logistic_fit <- loans_wf %>%
  fit(data = loans_training)
loans_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
##                        9.0680                        27.1336
##                   installment                  interest_rate
##                      -24.4930                        -2.7640
##                 annual_income              current_job_years
##                       -0.2113                         0.1001
##                debt_to_income             total_credit_lines
##                       -0.3291                         0.2564
##          years_credit_history        loan_purpose_credit_card
##                       -0.1450                        -1.0496
##          loan_purpose_medical    loan_purpose_small_business
##                       -1.6486                         0.3135
## loan_purpose_home_improvement         application_type_joint
##                        0.1656                        -0.6748
##                 term_five_year             homeownership_rent
##                       -15.5329                        -0.6933
##              homeownership_own          missed_payment_2_yr_no
##                        -0.5437                         0.1715
##          history_bankruptcy_no           history_tax_liens_no
##                        -0.1943                        -0.6532
##
## Degrees of Freedom: 3082 Total (i.e. Null);  3063 Residual
## Null Deviance:       4071
## Residual Deviance: 773.9     AIC: 813.9
```

```
# Exploring our Trained Model
# Extract the trained model from our workflow fit
loans_trained_model <- loans_logistic_fit %>%
  pull_workflow_fit()
loans_trained_model
```

```
## parsnip model object
##
## Fit time:  30ms
##
## Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)
##
## Coefficients:
##                    (Intercept)                    loan_amount
##                         9.0680                        27.1336
##                    installment                  interest_rate
##                       -24.4930                        -2.7640
##                  annual_income              current_job_years
##                        -0.2113                         0.1001
##                 debt_to_income             total_credit_lines
##                        -0.3291                         0.2564
##           years_credit_history        loan_purpose_credit_card
##                        -0.1450                        -1.0496
##           loan_purpose_medical     loan_purpose_small_business
##                        -1.6486                         0.3135
## loan_purpose_home_improvement          application_type_joint
##                         0.1656                        -0.6748
##                 term_five_year             homeownership_rent
##                       -15.5329                        -0.6933
##              homeownership_own          missed_payment_2_yr_no
##                        -0.5437                         0.1715
##          history_bankruptcy_no           history_tax_liens_no
##                        -0.1943                        -0.6532
##
## Degrees of Freedom: 3082 Total (i.e. Null);  3063 Residual
## Null Deviance:       4071
## Residual Deviance: 773.9     AIC: 813.9
```

```
# Variable Importance
vip(loans_trained_model)
```



```r
# Predicted Categories
predictions_categories <- predict(loans_logistic_fit,
                                  new_data = loans_test)
predictions_categories
```

```
## # A tibble: 1,027 x 1
##    .pred_class
##    <fct>
##  1 no
##  2 yes
##  3 no
##  4 no
##  5 no
##  6 yes
##  7 yes
##  8 yes
##  9 no
## 10 no
## # ... with 1,017 more rows
```

```r
# obtain the estimated probabilities for each category of our response variable
predictions_probabilities <- predict(loans_logistic_fit,
                                     new_data = loans_test,
                                     type = 'prob')
predictions_probabilities
```

```
## # A tibble: 1,027 x 2
##    .pred_yes .pred_no
##        <dbl>    <dbl>
##  1  0.00131   0.999
##  2  0.552     0.448
##  3  0.00318   0.997
##  4  0.000143  1.00
##  5  0.0738    0.926
##  6  0.992     0.00753
##  7  0.996     0.00405
##  8  0.989     0.0114
##  9  0.00570   0.994
## 10  0.00101   0.999
## # ... with 1,017 more rows
```

```
# combine the results from the Predicted Categories and  the estimated probabilities with the true resp
test_results <- loans_test %>%
  select(loan_default) %>%
  bind_cols(predictions_categories) %>%
  bind_cols(predictions_probabilities)
test_results
```

```
## # A tibble: 1,027 x 4
##    loan_default .pred_class .pred_yes .pred_no
##    <fct>        <fct>           <dbl>    <dbl>
##  1 no           no           0.00131   0.999
##  2 yes          yes          0.552     0.448
##  3 no           no           0.00318   0.997
##  4 no           no           0.000143  1.00
##  5 no           no           0.0738    0.926
##  6 yes          yes          0.992     0.00753
##  7 yes          yes          0.996     0.00405
##  8 yes          yes          0.989     0.0114
##  9 no           no           0.00570   0.994
## 10 no           no           0.00101   0.999
## # ... with 1,017 more rows
```

```
#6. Evaluate model performance on the test set by plotting an ROC curve using `autoplot()` and calculat
#------------------------------------------------------------------------------------------
# Evaluate the logistic regression model
#------------------------------------------------------------------------------------------
# Exploring Performance Metrics
# Confusion Matrix
conf_mat(test_results,
         truth = loan_default,
         estimate = .pred_class)
```

```
##           Truth
## Prediction yes  no
##        yes 354  16
##        no   28 629
```

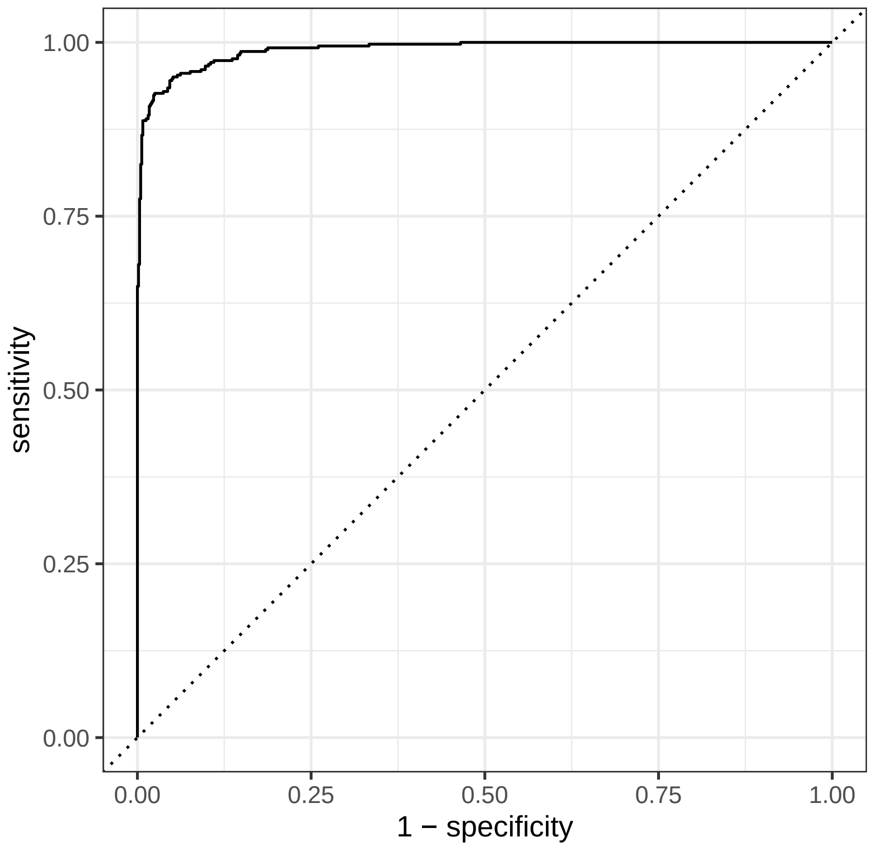
```
# F1 Score
f_meas(test_results,
       truth = loan_default,
       estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 f_meas  binary         0.941
```

```
# ROC Curve
roc_curve(test_results,
          truth = loan_default,
          estimate = .pred_yes)
```

```
## # A tibble: 1,029 x 3
##    .threshold specificity sensitivity
##         <dbl>       <dbl>       <dbl>
##  1 -Inf           0                 1
##  2    1.16e-9     0                 1
##  3    2.08e-9     0.00155           1
##  4    2.67e-9     0.00310           1
##  5    1.64e-8     0.00465           1
##  6    4.58e-8     0.00620           1
##  7    6.43e-8     0.00775           1
##  8    1.18e-7     0.00930           1
##  9    1.70e-7     0.0109            1
## 10    1.95e-7     0.0124            1
## # ... with 1,019 more rows
```

```
# plot the ROC Curve
roc_curve(test_results,
          truth = loan_default,
          estimate = .pred_yes) %>%
  autoplot()
```



```r
# Area Under the ROC Curve
roc_auc(test_results,
        truth = loan_default, .pred_yes)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.989
```

```r
# Creating Custom Metric Sets
# calculate the accuracy and F1 from my results data frame
my_metrics <- metric_set(accuracy,
                         f_meas)
my_metrics(test_results,
           truth = loan_default,
           estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.957
## 2 f_meas   binary         0.941
```

```r
# Automating the Process
last_fit_model <- loans_wf %>%
  last_fit(split = loans_split)
last_fit_model
```

```
## # Resampling results
## # Monte Carlo cross-validation (0.75/0.25) with 1 resamples
## # A tibble: 1 x 6
##   splits        id             .metrics     .notes      .predictions      .workflow
##   <list>        <chr>          <list>       <list>      <list>            <list>
## 1 <split [3.1K~ train/test ~ <tibble [2 ~ <tibble [0~ <tibble [1,027 ~ <workflo~
```

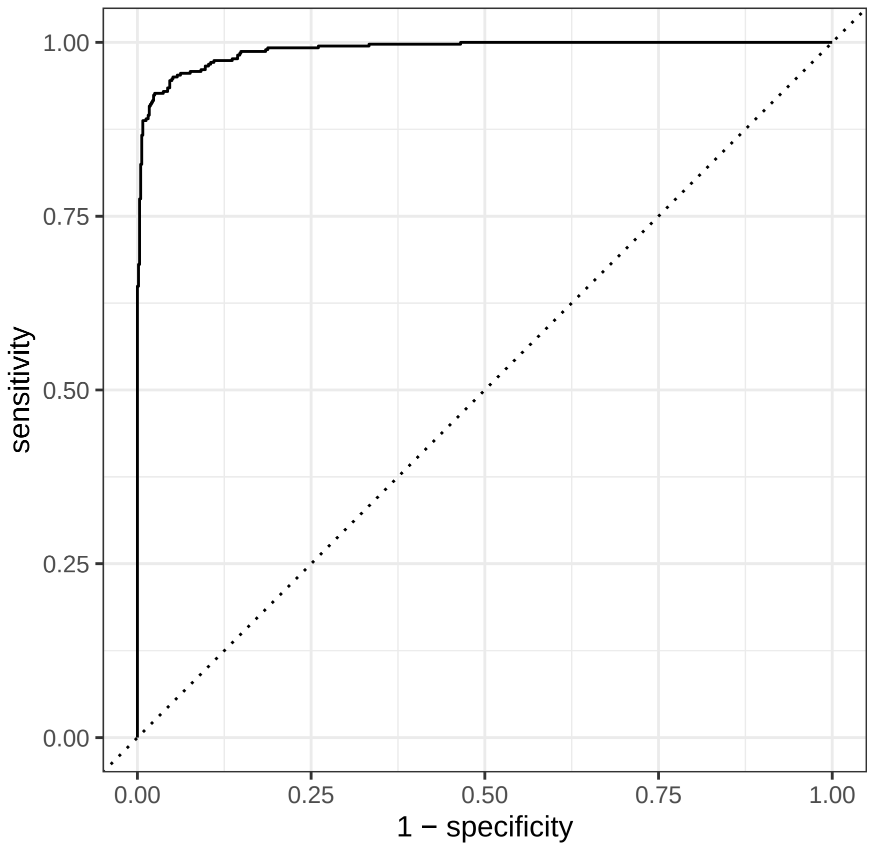
```
#obtain the metrics on the test set (accuracy and roc_auc by default)
last_fit_model %>%
  collect_metrics()
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.957
## 2 roc_auc  binary         0.989
```

```
# obtain a data frame with test set results
last_fit_results <- last_fit_model %>%
  collect_predictions()
last_fit_results
```

```
## # A tibble: 1,027 x 6
##    id               .pred_yes .pred_no  .row .pred_class loan_default
##    <chr>                <dbl>    <dbl> <int> <fct>       <fct>
##  1 train/test split  0.00131   0.999       3 no          no
##  2 train/test split  0.552     0.448       4 yes         yes
##  3 train/test split  0.00318   0.997       5 no          no
##  4 train/test split  0.000143  1.00       10 no          no
##  5 train/test split  0.0738    0.926      28 no          no
##  6 train/test split  0.992     0.00753    33 yes         yes
##  7 train/test split  0.996     0.00405    34 yes         yes
##  8 train/test split  0.989     0.0114     48 yes         yes
##  9 train/test split  0.00570   0.994      52 no          no
## 10 train/test split  0.00101   0.999      69 no          no
## # ... with 1,017 more rows
```

```
# make an ROC plot
last_fit_results %>%
  roc_curve(truth = loan_default,
            estimate = .pred_yes) %>%
  autoplot()
```



## Model 2

```
#------------------------------------------------------------------------------------
#3. Specify a `parsnip` model object
#------------------------------------------------------------------------------------
# Model Specification
#------------------------------------------------------------------------------------
# Decision Trees
#------------------------------------------------------------------------------------
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
```

```r
#4. Package your recipe and model into a workflow
#---------------------------------------------------------------------------------------
# Create a Workflow for decission tree
#---------------------------------------------------------------------------------------
tree_workflow <- workflow() %>%
  add_model(tree_model) %>%
  add_recipe(loans_recipe)
tree_workflow
```

```
## == 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 = tune()
##   tree_depth = tune()
##   min_n = tune()
##
## Computational engine: rpart
```

```r
# Create a grid of hyperparameter values to test
tree_grid <- grid_regular(cost_complexity(),
                          tree_depth(),
                          min_n(),
                          levels = 3)

# View grid
tree_grid
```

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

```r
# Create folds for cross validation on the training data set
## These will be used to tune model hyperparameters
set.seed(4500)
loans_folds <- vfold_cv(loans_training, v = 5)
loans_folds
```

```
## #  5-fold cross-validation
## # A tibble: 5 x 2
##   splits             id
##   <list>             <chr>
## 1 <split [2.5K/617]> Fold1
## 2 <split [2.5K/617]> Fold2
## 3 <split [2.5K/617]> Fold3
## 4 <split [2.5K/616]> Fold4
## 5 <split [2.5K/616]> Fold5
```

```r
## Tune decision tree workflow
set.seed(4500)
tree_tuning <- tree_workflow %>%
  tune_grid(resamples = loans_folds,
            grid = tree_grid)
tree_tuning
```

```
## # Tuning results
## # 5-fold cross-validation
## # A tibble: 5 x 4
##   splits             id    .metrics          .notes
##   <list>             <chr> <list>            <list>
## 1 <split [2.5K/617]> Fold1 <tibble [54 x 7]> <tibble [0 x 1]>
## 2 <split [2.5K/617]> Fold2 <tibble [54 x 7]> <tibble [0 x 1]>
## 3 <split [2.5K/617]> Fold3 <tibble [54 x 7]> <tibble [0 x 1]>
## 4 <split [2.5K/616]> Fold4 <tibble [54 x 7]> <tibble [0 x 1]>
## 5 <split [2.5K/616]> Fold5 <tibble [54 x 7]> <tibble [0 x 1]>
```

```r
## Show the top 5 best models based on roc_auc metric
tree_tuning %>% show_best('roc_auc')## Select best model based on roc_auc
```

```
## # A tibble: 5 x 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     0.964     5 0.00299
## 2    0.00000316           15    40 roc_auc binary     0.964     5 0.00299
## 3    0.0000000001         15    21 roc_auc binary     0.964     5 0.00357
## 4    0.00000316           15    21 roc_auc binary     0.964     5 0.00357
## 5    0.0000000001          8    40 roc_auc binary     0.959     5 0.00305
## # ... with 1 more variable: .config <fct>
```

```r
best_tree <- tree_tuning %>%
  select_best(metric = 'roc_auc')

# View the best tree parameters
best_tree
```

```
## # A tibble: 1 x 4
##   cost_complexity tree_depth min_n .config
##             <dbl>      <int> <int> <fct>
## 1    0.0000000001         15    40 Model25
```

```
# Finalize Workflow
final_tree_workflow <- tree_workflow %>%
  finalize_workflow(best_tree)
final_tree_workflow
```

```
## == 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
```

```
#5. Fit your workflow to the training data
#-------------------------------------------------------------------------------
# Fit the Model to the decission tree model
#-------------------------------------------------------------------------------
# Fit the workflow to the training data
tree_wf_fit <- final_tree_workflow %>%
  fit(data = loans_training)
tree_wf_fit
```

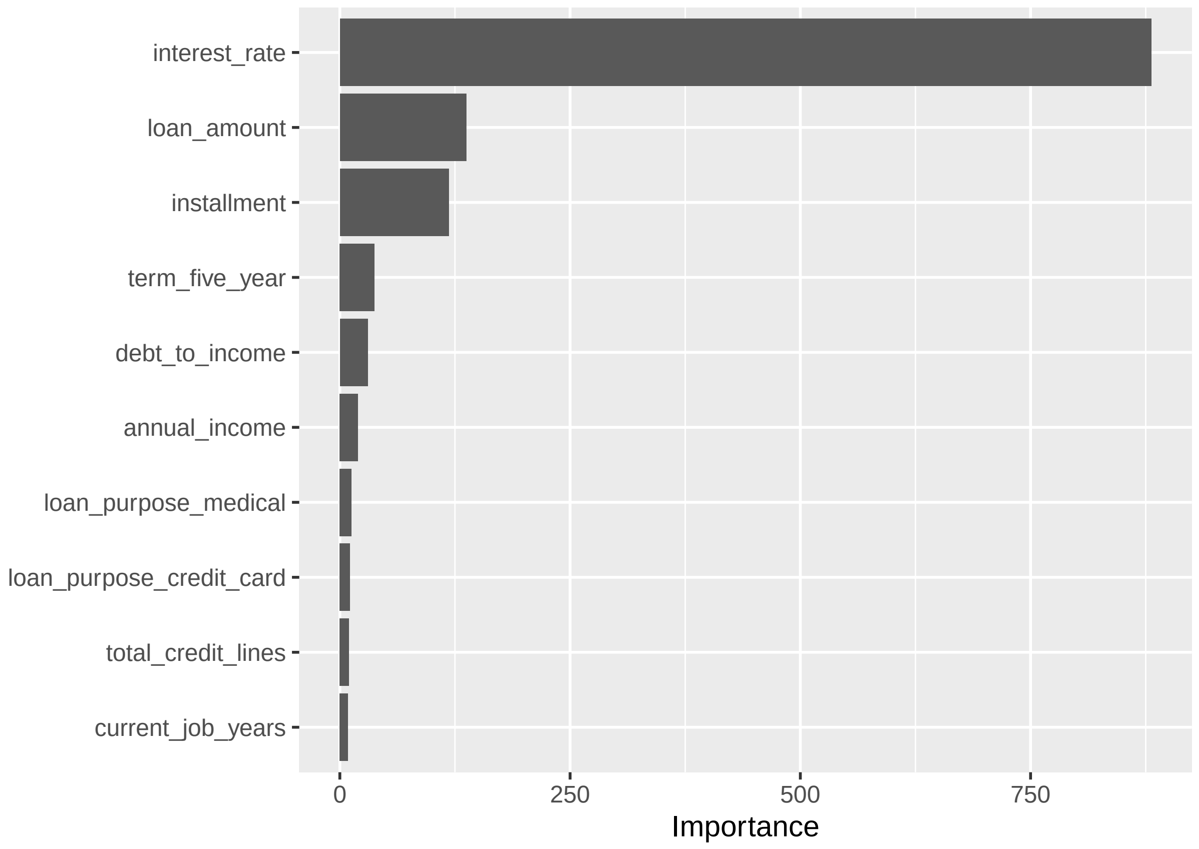
```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: decision_tree()
##
## -- Preprocessor ----------------------------------------------------------------
## 3 Recipe Steps
##
## * step_YeoJohnson()
## * step_normalize()
## * step_dummy()
##
## -- Model -----------------------------------------------------------------------
## n= 3083
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##    1) root 3083 1148 no (0.37236458 0.62763542)
##      2) interest_rate>=0.691997 702    0 yes (1.00000000 0.00000000) *
##      3) interest_rate< 0.691997 2381  446 no (0.18731625 0.81268375)
##        6) interest_rate>=-0.3606202 1287  446 no (0.34654235 0.65345765)
##         12) term_five_year>=0.5 473  234 yes (0.50528541 0.49471459)
##           24) installment>=-0.8083194 449  210 yes (0.53229399 0.46770601)
##             48) loan_amount< 1.59353 411  177 yes (0.56934307 0.43065693)
##               96) installment>=0.9637937 64    5 yes (0.92187500 0.07812500) *
##               97) installment< 0.9637937 347  172 yes (0.50432277 0.49567723)
##                194) loan_amount< 0.5606189 233   85 yes (0.63519313 0.36480687)
##                  388) installment>=0.2022886 43    0 yes (1.00000000 0.00000000) *
##                  389) installment< 0.2022886 190   85 yes (0.55263158 0.44736842)
##                    778) loan_amount< -0.327277 36    2 yes (0.94444444 0.05555556) *
##                    779) loan_amount>=-0.327277 154   71 no (0.46103896 0.53896104)
##                     1558) installment>=-0.2936345 100   38 yes (0.62000000 0.38000000)
##                       3116) loan_amount< 0.3146527 77   16 yes (0.79220779 0.20779221) *
##                       3117) loan_amount>=0.3146527 23    1 no (0.04347826 0.95652174) *
##                     1559) installment< -0.2936345 54    9 no (0.16666667 0.83333333) *
##                195) loan_amount>=0.5606189 114   27 no (0.23684211 0.76315789)
##                  390) installment>=0.5357504 69   26 no (0.37681159 0.62318841)
##                    780) loan_amount< 1.012413 25    3 yes (0.88000000 0.12000000) *
##                    781) loan_amount>=1.012413 44    4 no (0.09090909 0.90909091) *
##                  391) installment< 0.5357504 45    1 no (0.02222222 0.97777778) *
##             49) loan_amount>=1.59353 38    5 no (0.13157895 0.86842105) *
##           25) installment< -0.8083194 24    0 no (0.00000000 1.00000000) *
##         13) term_five_year< 0.5 814  207 no (0.25429975 0.74570025)
##           26) interest_rate< -0.3269675 13    0 yes (1.00000000 0.00000000) *
##           27) interest_rate>=-0.3269675 801  194 no (0.24219725 0.75780275)
##             54) loan_purpose_medical>=0.5 128   57 no (0.44531250 0.55468750)
##              108) debt_to_income>=-0.03797593 60   20 yes (0.66666667 0.33333333)
##                216) annual_income< 0.1502221 37    8 yes (0.78378378 0.21621622) *
##                217) annual_income>=0.1502221 23   11 no (0.47826087 0.52173913) *
##              109) debt_to_income< -0.03797593 68   17 no (0.25000000 0.75000000) *
##             55) loan_purpose_medical< 0.5 673  137 no (0.20356612 0.79643388)
##              110) loan_purpose_credit_card>=0.5 146   55 no (0.37671233 0.62328767)
##                220) loan_amount< -0.9609695 48   19 yes (0.60416667 0.39583333)
##                  440) interest_rate< 0.310133 30    8 yes (0.73333333 0.26666667) *
##                  441) interest_rate>=0.310133 18    7 no (0.38888889 0.61111111) *
##                221) loan_amount>=-0.9609695 98   26 no (0.26530612 0.73469388)
##                  442) installment>=0.08568682 50   22 no (0.44000000 0.56000000)
##                    884) debt_to_income>=0.1172241 30   12 yes (0.60000000 0.40000000) *
##                    885) debt_to_income< 0.1172241 20    4 no (0.20000000 0.80000000) *
##                  443) installment< 0.08568682 48    4 no (0.08333333 0.91666667) *
##              111) loan_purpose_credit_card< 0.5 527   82 no (0.15559772 0.84440228)
##                222) debt_to_income>=0.9236173 81   24 no (0.29629630 0.70370370) *
##
## ...
## and 10 more lines.
```

```
# Exploring our Trained Model
# Extract the trained model from our workflow fit
tree_fit <- tree_wf_fit %>%
  pull_workflow_fit()
tree_fit
```

```
## parsnip model object
##
```

```
## Fit time:  80ms
## n= 3083
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##    1) root 3083 1148 no (0.37236458 0.62763542)
##      2) interest_rate>=0.691997 702    0 yes (1.00000000 0.00000000) *
##      3) interest_rate< 0.691997 2381  446 no (0.18731625 0.81268375)
##        6) interest_rate>=-0.3606202 1287  446 no (0.34654235 0.65345765)
##         12) term_five_year>=0.5 473  234 yes (0.50528541 0.49471459)
##           24) installment>=-0.8083194 449  210 yes (0.53229399 0.46770601)
##             48) loan_amount< 1.59353 411  177 yes (0.56934307 0.43065693)
##               96) installment>=0.9637937 64    5 yes (0.92187500 0.07812500) *
##               97) installment< 0.9637937 347  172 yes (0.50432277 0.49567723)
##                194) loan_amount< 0.5606189 233   85 yes (0.63519313 0.36480687)
##                  388) installment>=0.2022886 43    0 yes (1.00000000 0.00000000) *
##                  389) installment< 0.2022886 190   85 yes (0.55263158 0.44736842)
##                    778) loan_amount< -0.327277 36    2 yes (0.94444444 0.05555556) *
##                    779) loan_amount>=-0.327277 154   71 no (0.46103896 0.53896104)
##                     1558) installment>=-0.2936345 100   38 yes (0.62000000 0.38000000)
##                       3116) loan_amount< 0.3146527 77   16 yes (0.79220779 0.20779221) *
##                       3117) loan_amount>=0.3146527 23    1 no (0.04347826 0.95652174) *
##                     1559) installment< -0.2936345 54    9 no (0.16666667 0.83333333) *
##                195) loan_amount>=0.5606189 114   27 no (0.23684211 0.76315789)
##                  390) installment>=0.5357504 69   26 no (0.37681159 0.62318841)
##                    780) loan_amount< 1.012413 25    3 yes (0.88000000 0.12000000) *
##                    781) loan_amount>=1.012413 44    4 no (0.09090909 0.90909091) *
##                  391) installment< 0.5357504 45    1 no (0.02222222 0.97777778) *
##             49) loan_amount>=1.59353 38    5 no (0.13157895 0.86842105) *
##           25) installment< -0.8083194 24    0 no (0.00000000 1.00000000) *
##         13) term_five_year< 0.5 814  207 no (0.25429975 0.74570025)
##           26) interest_rate< -0.3269675 13    0 yes (1.00000000 0.00000000) *
##           27) interest_rate>=-0.3269675 801  194 no (0.24219725 0.75780275)
##             54) loan_purpose_medical>=0.5 128   57 no (0.44531250 0.55468750)
##              108) debt_to_income>=-0.03797593 60   20 yes (0.66666667 0.33333333)
##                216) annual_income< 0.1502221 37    8 yes (0.78378378 0.21621622) *
##                217) annual_income>=0.1502221 23   11 no (0.47826087 0.52173913) *
##              109) debt_to_income< -0.03797593 68   17 no (0.25000000 0.75000000) *
##             55) loan_purpose_medical< 0.5 673  137 no (0.20356612 0.79643388)
##              110) loan_purpose_credit_card>=0.5 146   55 no (0.37671233 0.62328767)
##                220) loan_amount< -0.9609695 48   19 yes (0.60416667 0.39583333)
##                  440) interest_rate< 0.310133 30    8 yes (0.73333333 0.26666667) *
##                  441) interest_rate>=0.310133 18    7 no (0.38888889 0.61111111) *
##                221) loan_amount>=-0.9609695 98   26 no (0.26530612 0.73469388)
##                  442) installment>=0.08568682 50   22 no (0.44000000 0.56000000)
##                    884) debt_to_income>=0.1172241 30   12 yes (0.60000000 0.40000000) *
##                    885) debt_to_income< 0.1172241 20    4 no (0.20000000 0.80000000) *
##                  443) installment< 0.08568682 48    4 no (0.08333333 0.91666667) *
##              111) loan_purpose_credit_card< 0.5 527   82 no (0.15559772 0.84440228)
##                222) debt_to_income>=0.9236173 81   24 no (0.29629630 0.70370370) *
##                223) debt_to_income< 0.9236173 446   58 no (0.13004484 0.86995516)
##                  446) years_credit_history< -0.3443376 184   34 no (0.18478261 0.81521739)
##                    892) current_job_years>=0.004774538 61   18 no (0.29508197 0.70491803)
##                         1784) annual_income< 0.3357285 46   18 no (0.39130435 0.60869565)
##                           3568) homeownership_rent>=0.5 28   13 yes (0.53571429 0.46428571) *
##                           3569) homeownership_rent< 0.5 18    3 no (0.16666667 0.83333333) *
##                         1785) annual_income>=0.3357285 15    0 no (0.00000000 1.00000000) *
##                        893) current_job_years< 0.004774538 123   16 no (0.13008130 0.86991870) *
##                      447) years_credit_history>=-0.3443376 262   24 no (0.09160305 0.90839695) *
##        7) interest_rate< -0.3606202 1094    0 no (0.00000000 1.00000000) *
```

```r
# The variable importance scores from the model
vip(tree_fit)
```



```r
# Predicted Categories
tree_predictions_categories <- predict(tree_wf_fit,
                                       new_data = loans_test)
tree_predictions_categories
```

```
## # A tibble: 1,027 x 1
##    .pred_class
##    <fct>
##  1 no
##  2 no
##  3 no
##  4 no
##  5 no
##  6 yes
##  7 yes
##  8 yes
##  9 no
## 10 no
## # ... with 1,017 more rows
```

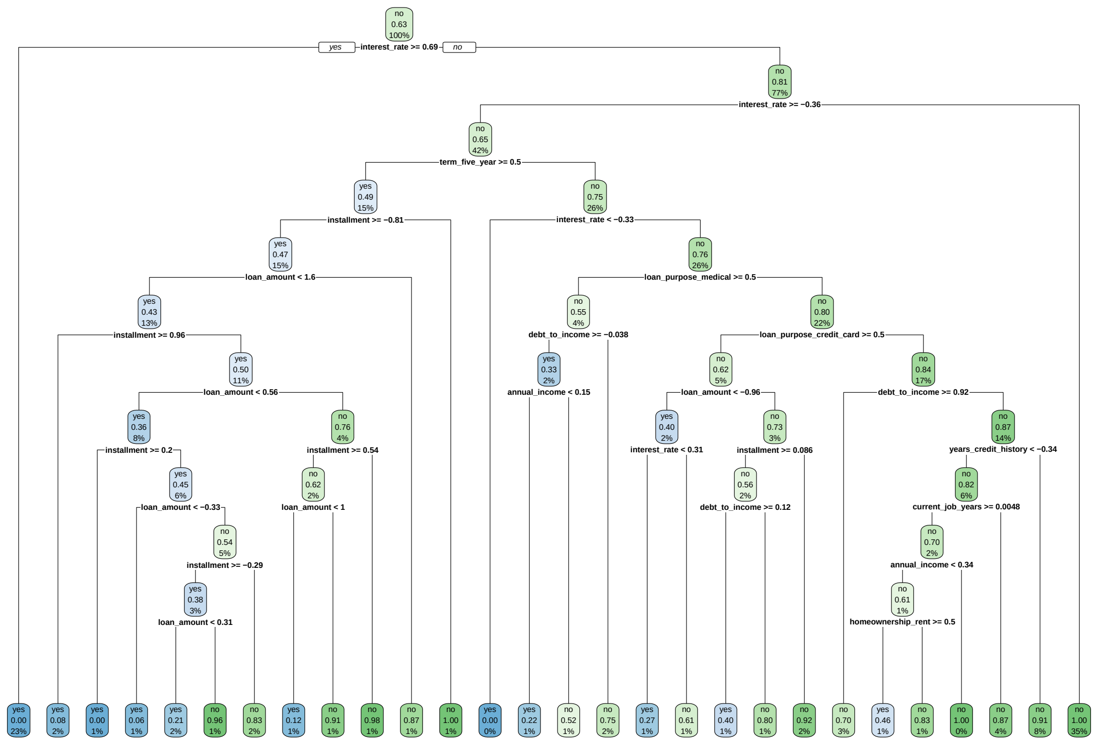
```
# obtain the estimated probabilities for each category of our response variable
tree_predictions_probabilities <- predict(tree_wf_fit,
                                          new_data = loans_test,
                                          type = 'prob')
tree_predictions_probabilities
```

```
## # A tibble: 1,027 x 2
##    .pred_yes .pred_no
##        <dbl>    <dbl>
##  1     0        1
##  2     0.25     0.75
##  3     0        1
##  4     0.132    0.868
##  5     0.389    0.611
##  6     1        0
##  7     1        0
##  8     1        0
##  9     0        1
## 10     0        1
## # ... with 1,017 more rows
```

```
# combine the results from the Predicted Categories and  the estimated probabilities with the true resp
tree_test_results <- loans_test %>%
  select(loan_default) %>%
  bind_cols(tree_predictions_categories) %>%
  bind_cols(tree_predictions_probabilities)
tree_test_results
```

```
## # A tibble: 1,027 x 4
##    loan_default .pred_class .pred_yes .pred_no
##    <fct>        <fct>           <dbl>    <dbl>
##  1 no           no              0        1
##  2 yes          no              0.25     0.75
##  3 no           no              0        1
##  4 no           no              0.132    0.868
##  5 no           no              0.389    0.611
##  6 yes          yes             1        0
##  7 yes          yes             1        0
##  8 yes          yes             1        0
##  9 no           no              0        1
## 10 no           no              0        1
## # ... with 1,017 more rows
```

```
# Decision Tree Plot
rpart.plot(tree_fit$fit,
           roundint = FALSE)
```



```
#6. Evaluate model performance on the test set by plotting an ROC curve using `autoplot()` and calculat
#-----------------------------------------------------------------------------------
# Evaluate the decission tree model
#-----------------------------------------------------------------------------------
# Exploring Performance Metrics
# Confusion Matrix
conf_mat(tree_test_results,
         truth = loan_default,
         estimate = .pred_class)
```

```
##            Truth
## Prediction yes  no
##        yes 330  28
##        no   52 617
```

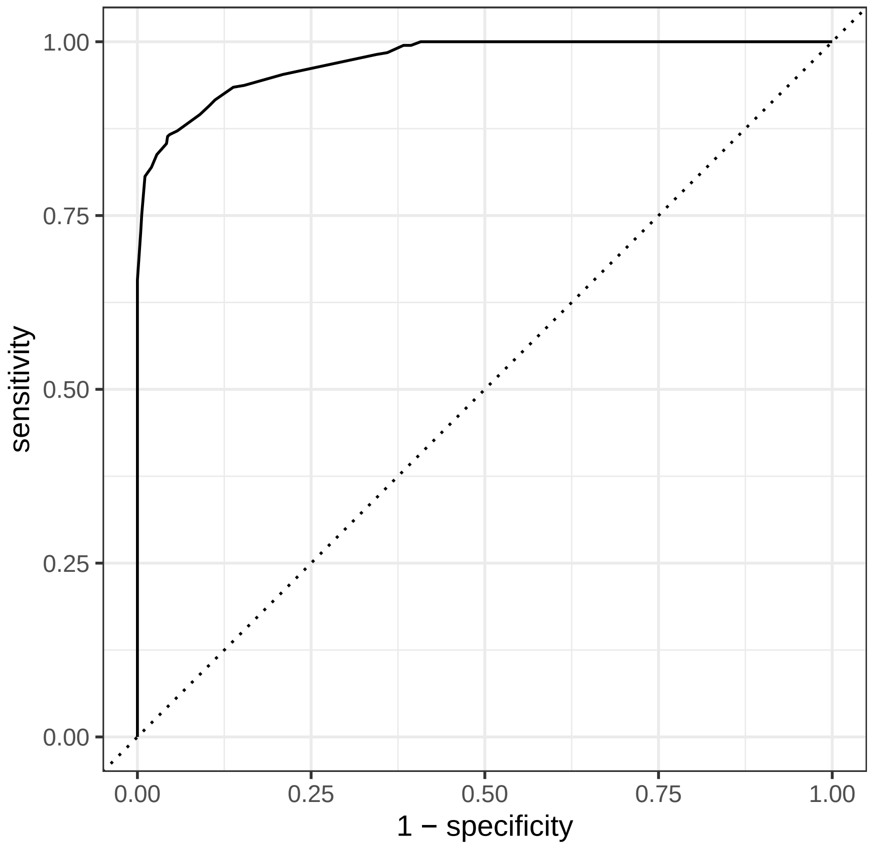
```
# F1 Score
f_meas(tree_test_results,
       truth = loan_default,
       estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 f_meas  binary         0.892
```

```
# ROC Curve
roc_curve(tree_test_results,
          truth = loan_default,
          estimate = .pred_yes)
```

```
## # A tibble: 25 x 3
##    .threshold specificity sensitivity
##         <dbl>       <dbl>       <dbl>
##  1  -Inf            0           1
##  2     0            0           1
##  3     0.0222       0.592       1
##  4     0.0435       0.606       0.995
##  5     0.0833       0.617       0.995
##  6     0.0909       0.640       0.984
##  7     0.0916       0.656       0.982
##  8     0.130        0.791       0.953
##  9     0.132        0.847       0.937
## 10     0.167        0.862       0.935
## # ... with 15 more rows
```

```
# plot the ROC Curve
roc_curve(tree_test_results,
          truth = loan_default,
          estimate = .pred_yes) %>%
  autoplot()
```



```
# Area Under the ROC Curve
roc_auc(tree_test_results,
        truth = loan_default, .pred_yes)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.973
```

```r
# Creating Custom Metric Sets
# calculate the accuracy and F1 from my results data frame
my_metrics_tree <- metric_set(accuracy,
                              f_meas)
my_metrics_tree(tree_test_results,
                truth = loan_default,
                estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.922
## 2 f_meas   binary         0.892
```

```r
# Automating the Process
tree_last_fit <- final_tree_workflow %>%
  last_fit(loans_split)
tree_last_fit
```

```
## # Resampling results
## # Monte Carlo cross-validation (0.75/0.25) with 1 resamples
## # A tibble: 1 x 6
##   splits        id            .metrics     .notes      .predictions      .workflow
##   <list>        <chr>         <list>       <list>      <list>            <list>
## 1 <split [3.1K~ train/test ~ <tibble [2 ~ <tibble [0~ <tibble [1,027 ~ <workflo~
```

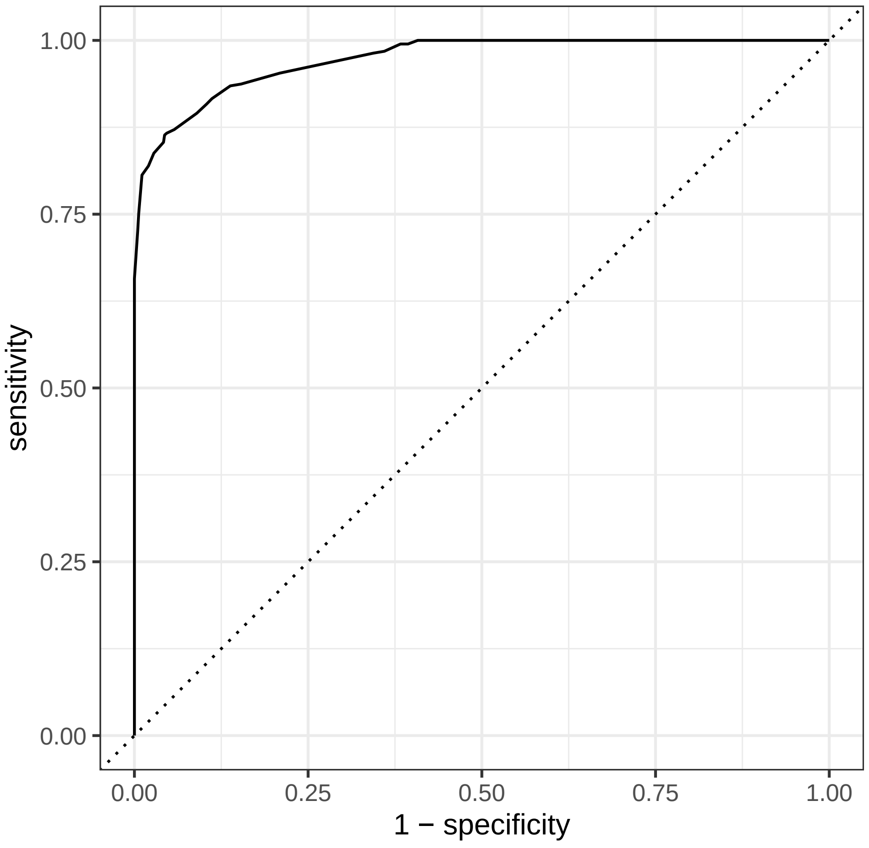
```r
#obtain the metrics on the test set (accuracy and roc_auc by default)
tree_last_fit %>%
  collect_metrics()
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.922
## 2 roc_auc  binary         0.973
```

```r
# obtain a data frame with test set results
tree_last_fit_results <- tree_last_fit %>%
  collect_predictions()
tree_last_fit_results
```

```
## # A tibble: 1,027 x 6
##    id               .pred_yes .pred_no  .row .pred_class loan_default
##    <chr>                <dbl>    <dbl> <int> <fct>       <fct>
##  1 train/test split     0        1         3 no          no
##  2 train/test split     0.25     0.75      4 no          yes
##  3 train/test split     0        1         5 no          no
##  4 train/test split     0.132    0.868    10 no          no
##  5 train/test split     0.389    0.611    28 no          no
##  6 train/test split     1        0        33 yes         yes
##  7 train/test split     1        0        34 yes         yes
##  8 train/test split     1        0        48 yes         yes
##  9 train/test split     0         1         52 no           no
## 10 train/test split     0         1         69 no           no
## # ... with 1,017 more rows
```

```
# plot the ROC curve to visualize test set performance of the tuned decision tree
tree_last_fit_results %>%
  roc_curve(truth  = loan_default,
            estimate = .pred_yes) %>%
  autoplot()
```



## Summary of Results [25 Points]

Write a summary of your overall findings and recommendations to the executives at the bank. Think of this section as your closing remarks of a presentation, where you summarize your key findings, model performance, and make recommendations to improve loan processes at the bank.

Your summary should include:

- Key findings from your data analysis. What were the things that stuck out for you in this section and why are they important?
- Your "best" classification model and an analysis of its performance.
  - In this section you should talk about the expected error of your model on future data.
  - You should discuss at least one performance metric, such as an F1 or ROC AUC for your model. However, you must explain the results in an intuitive, non-technical manner. Your audience in this case are executives at a bank with limited knowledge of machine learning.
- Your recommendations to the bank – how can financial losses from default be reduced?

**Summary**

The main thing that can be concluded by the data analysis part is that the number of the total number of the non default customer is higher than the loan default customer. Therefore, problem might occur if there is a special rare case for the loan default customer appears. Other than that, there is some predictors which would suppose to be better if provided by a set fix of value or factor instead of being the numeric type. For an instance, the interest rate which might be better to be replaced by type of loan product so that the difference between the interest rate and the loan default can be easily found since each loan product should usually has a fix interest rate. Also, information about loan collectibilty should be provided as well in the dataset because the relationship with the loan default is usually strong.

The classifications models, which are used to predict the loan default customer, are logistic regression and decission tree. Those two models are tested through the test set which were obtained from the first split at beginning. The number of row for the test set is 1027. Therefore, the two models are tested by using the same 1027 rows dataset.

Firstly, we will explore the performance thorugh the confusion metric. Respectively, the confusion metric of the first and the second model are such the follows:

- logistic regression: Truth Prediction yes no yes 354 16 no 28 629
- decission tree Truth Prediction yes no yes 330 28 no 52 617

According to the first confusion metric result, we can see that there are 983 rows out of 1027 have the right prediction. Specifically, in that 983 rows, there 354 rows of the customers loan default have the same value with the predicted value and there are 629 of the non default customers have the same value with the predicted value. However, there are 44 rows which are missed predicted since the actual values are different to the predicted value. Therefore, according to such confusion metric, we can conclude that the accuracy of this model is 95%.

HOwever, according to the second confusion metric, we can calculate that the accuracy of the model is 92%. Therefore, the accuracy of the second model, which is the decision tree model, based on the confusion metric, is not as good as the first model. Yet, the difference of the two wrong predicted values of the second model are quite signifinicant, which are 52 and 28. Therefore, the F1 score should be calculated in order to support the performance measurement of the second model.

According to the both accuracy, the expected error from the test set respectively of those two models are 0.043 and 0.078. In other words, the tendecy of the logistic regression model to yield the error result is 4.3% and the decision tree is 7.8%. Therefore, the logistic regression still has the smaller chance to gain error results while predicting whether the customer tend to be default or non default in loan payment.

Secondly, we explore the F1 score that serves as the performance metric which balances the missed predicted values including the customers which are wrongly predicted as the loan default or the non default status. Respectively, the F1 score of the first model and second model while tested by using the test set is such the follows:

- logistic regression: f_meas binary 0.9414894
- decission tree: f_meas binary 0.8918919

According to number of missed predicted value in the confusion metric of the first model above, the difference between false positive, which is the worng prediction for the default customer, and the false negative, which is the total number of the wrong prediction for the non default custemer, are not significantly different, using the accuracy 95% accuracy as the performance measurement is still okay. Even though, according to the F1 score, the performance value is still very good, which is 0.94, since the value is close to 1. Obviously, compared to the F1 score of the second model, the first model still has the better performance since the F1 score of the second model is 0.89 which is lowever then the F1 score of the losgistic regression model.

Thirdly, we explore the ROC curve to measure the model performance. According to the ROC Curve of the logistic regresion model, the plot between the sensitivity, which is the true positive fraction (the true “yes” of loan default), and the specificity, which is the false positive fraction, is close the the 0,1 point. The firt model is even better than the second model since based on the ROC curve, the first model seems to have the closer curve to the 0,1 point, which is on top left corner.

The last one, we explore the Area Under the ROC Curve. Respectively, the AU ROC of the Logistic model and the decision tree are shuch the follows:

- logistic regression: roc_auc binary 0.9894598
- decission tree: roc_auc binary 0.9729595

The AU ROC, which falls between 0.9 to 1, shows a good model performance. Since the AU ROC for the logistic regresion model is 0.9894598, which is very close to 1, then the model is classified as a very good model. The same as the decission tree model, the both models can be considered as the good model. HOwever, in overall, the logistic regression model is the better model to the classification for the loan default customer based on the given dataset.

According to all the result, I suggest the bank to pay attention more to the customers as the following criterias:

1. The customers who use the loan for credit card and medical purpose
2. The customers within 5 years term of loan
3. The customers who have worked for 8 years in the current job
4. The customers which have the loan amount above 10000
5. The customers which have installment above 490.00
6. The customers who live in rent houses

The above criterias shows that the customers have tendency to get the loan default status. Therefore, those kind of customers can be treated by restructring the loan account whether by extending the payment term and reducing the interest rate, or reverting the method of loan payment by paying principle loan amount and followed by interest payment, or vice versa.