Vignette

2023-12-05

Binary Classification Vignette

For the purpose of this vignette, we will use data from the National Institute of Diabetes and Digestive Kidney Disease.

Activity: Creating different models using binary classification algorithms

We will fit multiple models and compute basic classification accuracy measures in order to compare the models and see which one should be the final model chosen.

Prerequisites First we will start the setup by loading the required packages and data.

```
# load packages
library(readr)
library(vip)
```

```
##
## Attaching package: 'vip'

## The following object is masked from 'package:utils':
##
##     vi
```

```
library(naniar)
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.1.1 --

## v broom        1.0.5     v recipes      1.0.8
## v dials        1.2.0     v rsample      1.2.0
## v dplyr        1.1.3     v tibble       3.2.1
## v ggplot2      3.4.4     v tidyr        1.3.0
## v infer        1.0.5     v tune         1.1.2
## v modeldata    1.2.0     v workflows    1.1.3
## v parsnip      1.1.1     v workflowsets 1.0.1
## v purrr        1.0.2     v yardstick    1.2.0

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard()  masks scales::discard()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Use tidymodels_prefer() to resolve common conflicts.
```

```
library(ISLR)
library(ISLR2)
```

```
##
## Attaching package: 'ISLR2'

## The following objects are masked from 'package:ISLR':
##
##     Auto, Credit
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ----------------------- tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.3

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x scales::col_factor() masks readr::col_factor()
## x purrr::discard()     masks scales::discard()
## x dplyr::filter()      masks stats::filter()
## x stringr::fixed()     masks recipes::fixed()
## x dplyr::lag()         masks stats::lag()
## x yardstick::spec()    masks readr::spec()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library(modeldata)
library(ggthemes)
library(janitor)
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```r
library(kableExtra)
```

```
## Warning in !is.null(rmarkdown::metadata$output) && rmarkdown::metadata$output
## %in% : 'length(x) = 2 > 1' in coercion to 'logical(1)'
```

```
##
## Attaching package: 'kableExtra'
##
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```r
library(yardstick)
library(kknn)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
library(themis)
library(dplyr)
library(ggplot2)
library(scales)
library(rpart.plot)
```

```
## Loading required package: rpart
##
## Attaching package: 'rpart'
##
## The following object is masked from 'package:dials':
##
##     prune
```

```r
library(discrim)
```

```
##
## Attaching package: 'discrim'
##
## The following object is masked from 'package:dials':
##
##     smoothness
```

```r
library(klaR)
```

```
## Loading required package: MASS
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:ISLR2':
##
##     Boston
```

```
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(plotly)
```

```
##
## Attaching package: 'plotly'
##
## The following object is masked from 'package:MASS':
##
##     select
##
## The following object is masked from 'package:ggplot2':
##
##     last_plot
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:plotly':
##
##     slice
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
library(recipes)
library(ROSE)
```

```
## Loaded ROSE 0.0-4
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(reticulate)
tidymodels_prefer()

# read data
db <- read_csv("data/diabetes.csv")
```

```
## Rows: 768 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (9): Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, D...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
db <- read_csv("data/diabetes.csv")
```

```
## Rows: 768 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (9): Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, D...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#bank_df$Exited_num <- as.numeric(bank_df$Exited) # Convert Exited to numeric variable
diabetes_numeric <- db %>%
  select_if(is.numeric) # Select only numeric columns
cor_matrix <- cor(diabetes_numeric) # Compute correlation matrix
```

```
# covert survived and pclass into factors
diabetes_numeric$Outcome <- as.factor(diabetes_numeric$Outcome)
# sort the data frame by survived, so the yes will be on top
db_sort <- db%>% arrange(desc(Outcome))
```

Data Partitioning When data partitioning, we split the data where the training set is used to train the model and the test set is used to evaluate the performance of the model. Partitions are computed at random.

First we will do cross-validation and data splitting.
We then partition the diabetes data into training and test sets.

```
set.seed(3435)
db_split <- initial_split(db, prop = 0.80,
                               strata = "Outcome")
db_train <- training(db_split)
```

```r
db_test <- testing(db_split)
db_fold <- vfold_cv(db_train,v=4)

# Data splitting
db$Outcome <- as.factor(db$Outcome)
set.seed(3435)
db_split <- initial_split(db, prop = 0.80, strata = "Outcome")
db_train <- training(db_split)
db_test <- testing(db_split)
db_fold <- vfold_cv(db_train, v = 6)
```

Recipe…

```r
# Recipe without step_upsample
db_recipe <- recipe(Outcome ~ ., data = db_train) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_predictors())

# Prepare the recipe
prep(db_recipe)%>% bake(new_data = db_train) %>%
  group_by(Outcome) %>%
  summarise(count = n())
```

```
## # A tibble: 2 x 2
##   Outcome count
##   <fct>   <int>
## 1 0         400
## 2 1         214
```

Binary Classification Algorithm #1 Logistic Regression

```r
db_train$Outcome <- factor(db_train$Outcome)

# Logistic regression
log_reg <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

db_log_wflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(db_recipe)

# Tune the model
db_tune_reg <- tune_grid(
  object = db_log_wflow,
  resamples = db_fold
)
```

```
## Warning: No tuning parameters have been detected, performance will be evaluated
## using the resamples with no tuning. Did you want to [tune()] parameters?
```

Logistic Model Fitting

```r
# Fit the model
log_fit <- fit(db_log_wflow, db_train)
```

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```r
# Extract predictions
log_preds <- augment(log_fit, new_data = db_train)

# Calculate ROC AUC directly using pROC
roc_curve <- roc(log_preds$Outcome, log_preds$.pred_0)
```

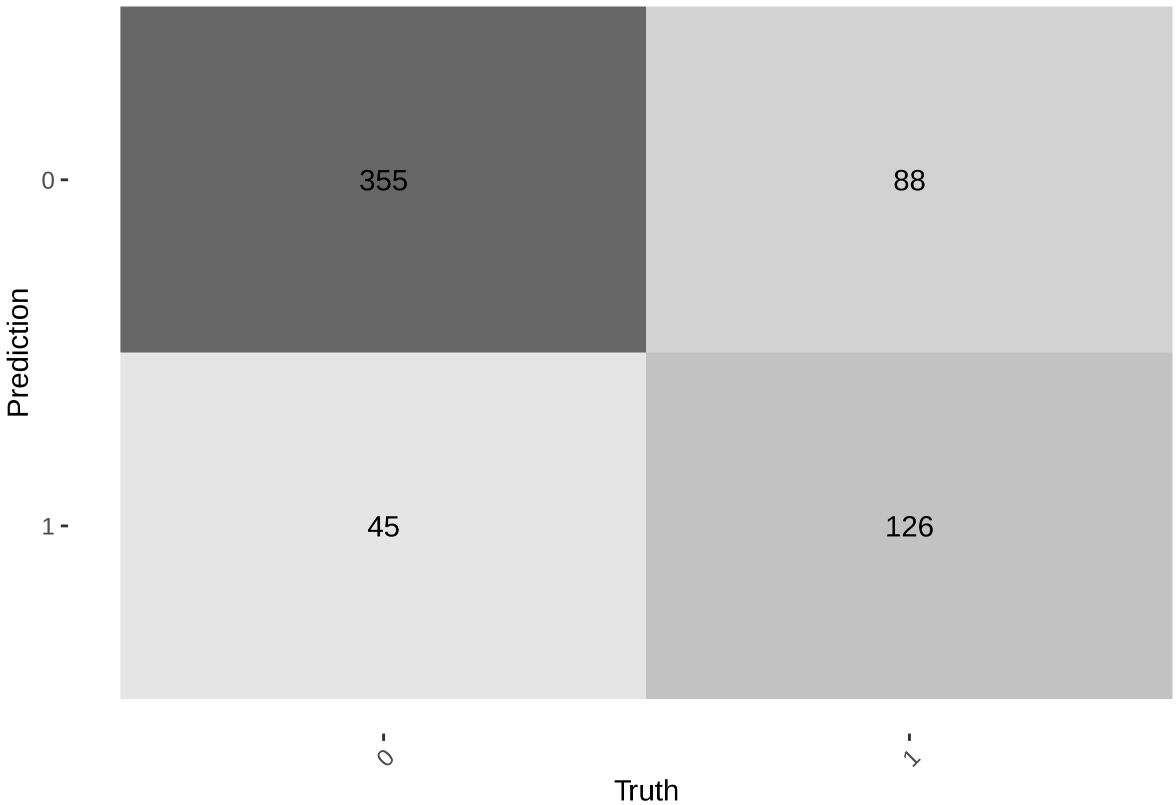
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

```r
roc_auc_value <- auc(roc_curve)
print(roc_auc_value)
```

```
## Area under the curve: 0.8433
```

```r
# Confusion Matrix
conf_matrix <- log_preds %>%
  conf_mat(truth = Outcome, estimate = .pred_class)

# Plot Confusion Matrix
conf_matrix %>% autoplot(type = 'heatmap') +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
```




```
# Show best tuning parameters
show_best(db_tune_reg, n = 1)
```

```
## Warning: No value of `metric` was given; metric 'roc_auc' will be used.

## # A tibble: 1 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 roc_auc binary     0.834     6 0.00921 Preprocessor1_Model1
```

Accuracy Measures

```
library(pROC)

# Extract predictions
log_preds <- augment(log_fit, new_data = db_train)

log_predictions <- augment(log_fit, new_data = db_test)
log_accuracy <- accuracy(log_predictions, truth = Outcome, estimate = .pred_class)
log_conf_matrix <- conf_mat(log_predictions, truth = Outcome, estimate = .pred_class)
```

We will then use the predictions and the observed classes to create a Confusion Matrix table.

```
library(pROC)

# Calculate ROC AUC directly using pROC
roc_curve <- roc(log_preds$Outcome, log_preds$.pred_0)
```

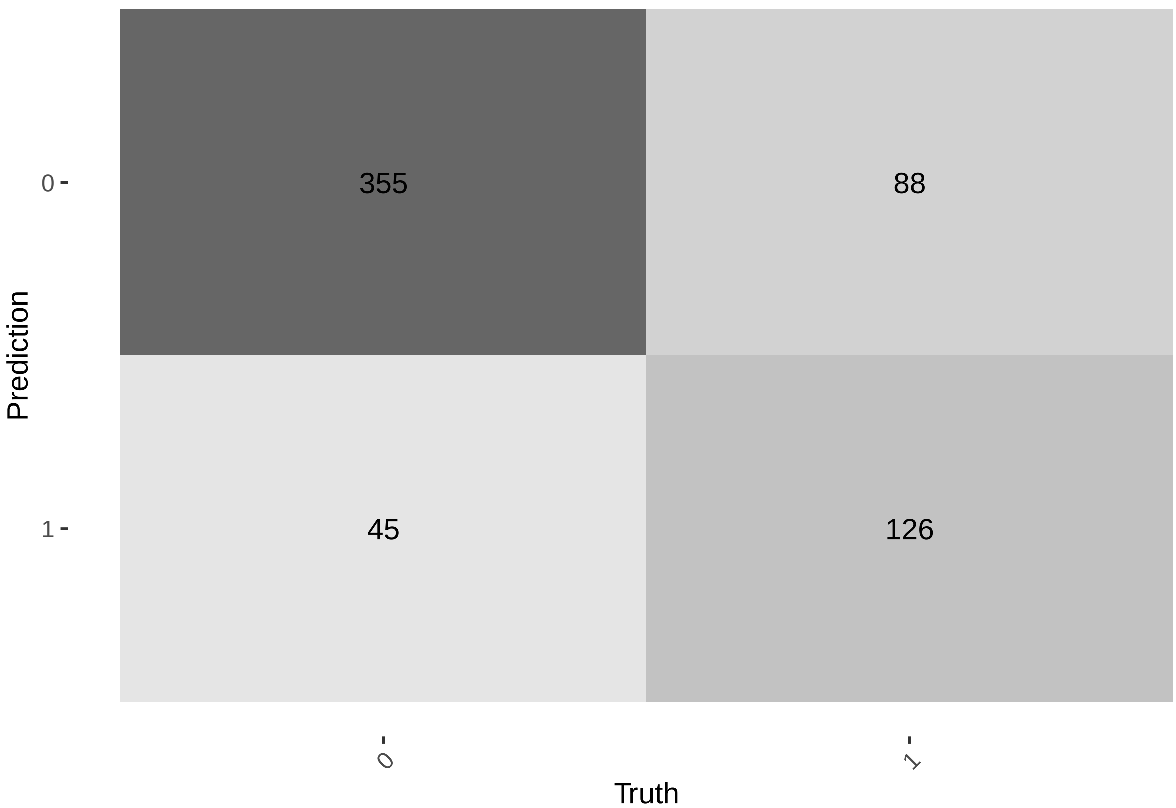
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

```
roc_auc_value <- auc(roc_curve)
print(roc_auc_value)
```

```
## Area under the curve: 0.8433
```

```
# Confusion Matrix
conf_matrix <- log_preds %>%
  conf_mat(truth = Outcome, estimate = .pred_class)

# Plot Confusion Matrix
conf_matrix %>% autoplot(type = 'heatmap') +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
```




```
# Show best tuning parameters
show_best(db_tune_reg, n = 1)
```

```
## Warning: No value of `metric` was given; metric 'roc_auc' will be used.
```

```
## # A tibble: 1 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 roc_auc binary     0.834     6 0.00921 Preprocessor1_Model1
```

Binary Classification Algorithm #2 K-Nearest Neighbors

```
# Recipe without step_upsample
db_recipe <- recipe(Outcome ~ ., data = db_train) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_predictors())

# Prepare the recipe
prep(db_recipe)%>% bake(new_data = db_train) %>%
  group_by(Outcome) %>%
  summarise(count = n())
```

```
## # A tibble: 2 x 2
##   Outcome count
##   <fct>   <int>
## 1 0         400
## 2 1         214
```

```
# Define the k-NN model
knn_model <- nearest_neighbor(neighbors = tune()) %>%
  set_engine("kknn") %>%
  set_mode("classification")

# Create a workflow with the recipe and model
db_knn_wflow <- workflow() %>%
  add_recipe(db_recipe) %>%  # Include the recipe
  add_model(knn_model)      # Include the model

# Grid for tuning
neighbors_grid <- grid_regular(neighbors(range = c(1, 10)), levels = 10)

# Tune the model
db_tune_knn <- tune_grid(
  object = db_knn_wflow,
  resamples = db_fold,
  grid = neighbors_grid
)

# Select the best model
best_knn_db <- select_best(
  db_tune_knn,
  metric = "roc_auc",
  neighbors
)
```

We will then use the predictions and observed classes to create a Confusion Matrix table.

```
library(pROC)

best_knn_wf <- finalize_workflow(db_knn_wflow, best_knn_db)
knn_fit <- fit(best_knn_wf, data = db_train)

# Extract predictions
```

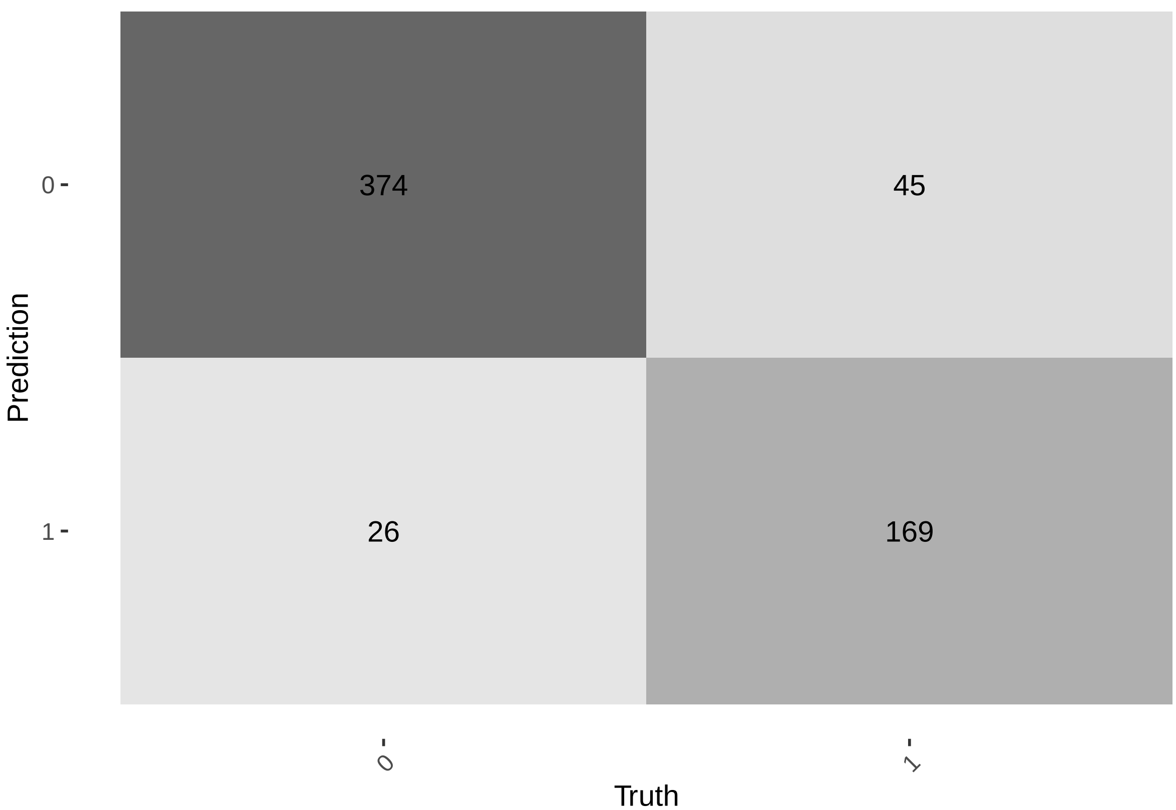
```
knn_preds <- augment(knn_fit, new_data = db_train)

# Calculate ROC AUC directly using pROC
roc_curve <- roc(knn_preds$Outcome, knn_preds$.pred_0)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls > cases
```

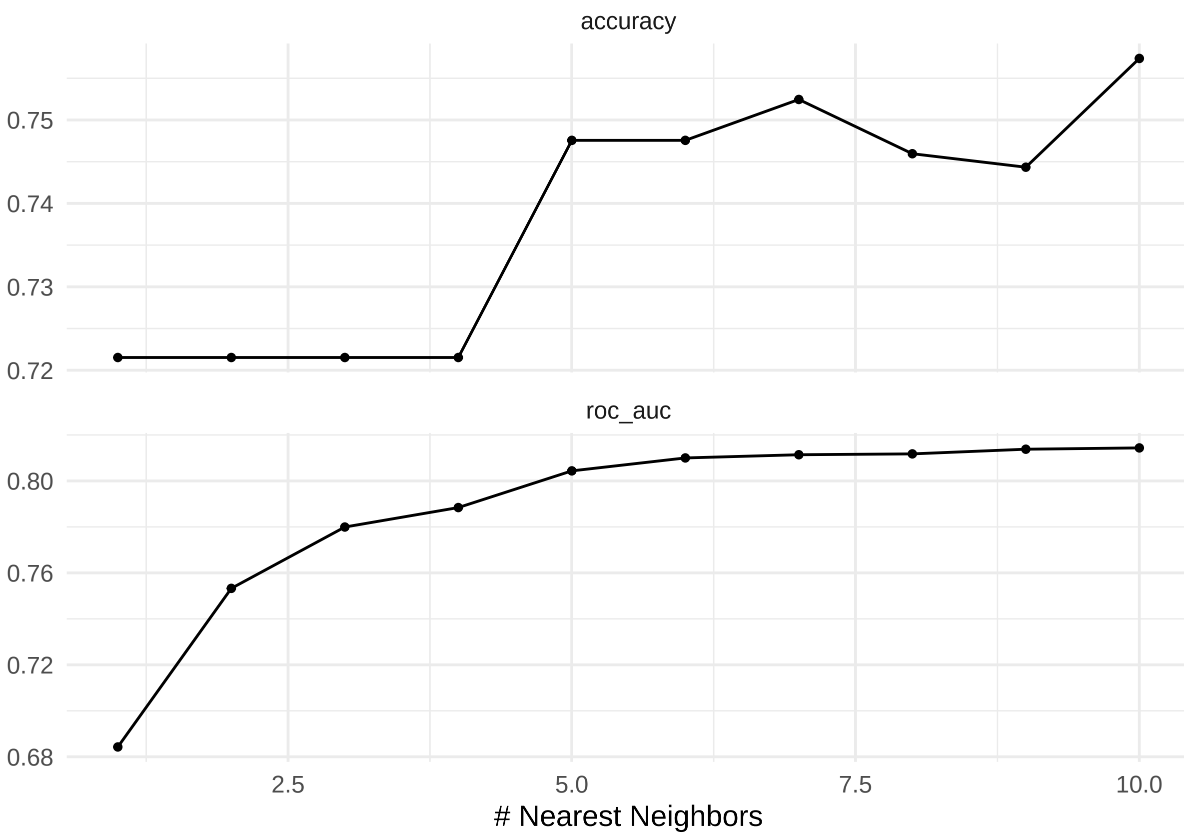
```
roc_auc_value <- auc(roc_curve)
print(roc_auc_value)
```

```
## Area under the curve: 0.9626
```

```
# Confusion Matrix
conf_matrix <- knn_preds %>%
  conf_mat(truth = Outcome, estimate = .pred_class)

# Plot Confusion Matrix
conf_matrix %>% autoplot(type = 'heatmap') +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
```




```
autoplot(db_tune_knn) + theme_minimal()
```



Binary Classification Algorithm #3 Boosted Tree Model

```
bt_class_spec <- boost_tree(mtry = tune(),
                            trees = tune(),
                            learn_rate = tune()) %>%
  set_engine("xgboost") %>%
  set_mode("classification")

bt_class_wf <- workflow() %>%
  add_model(bt_class_spec) %>%
  add_recipe(db_recipe)
```

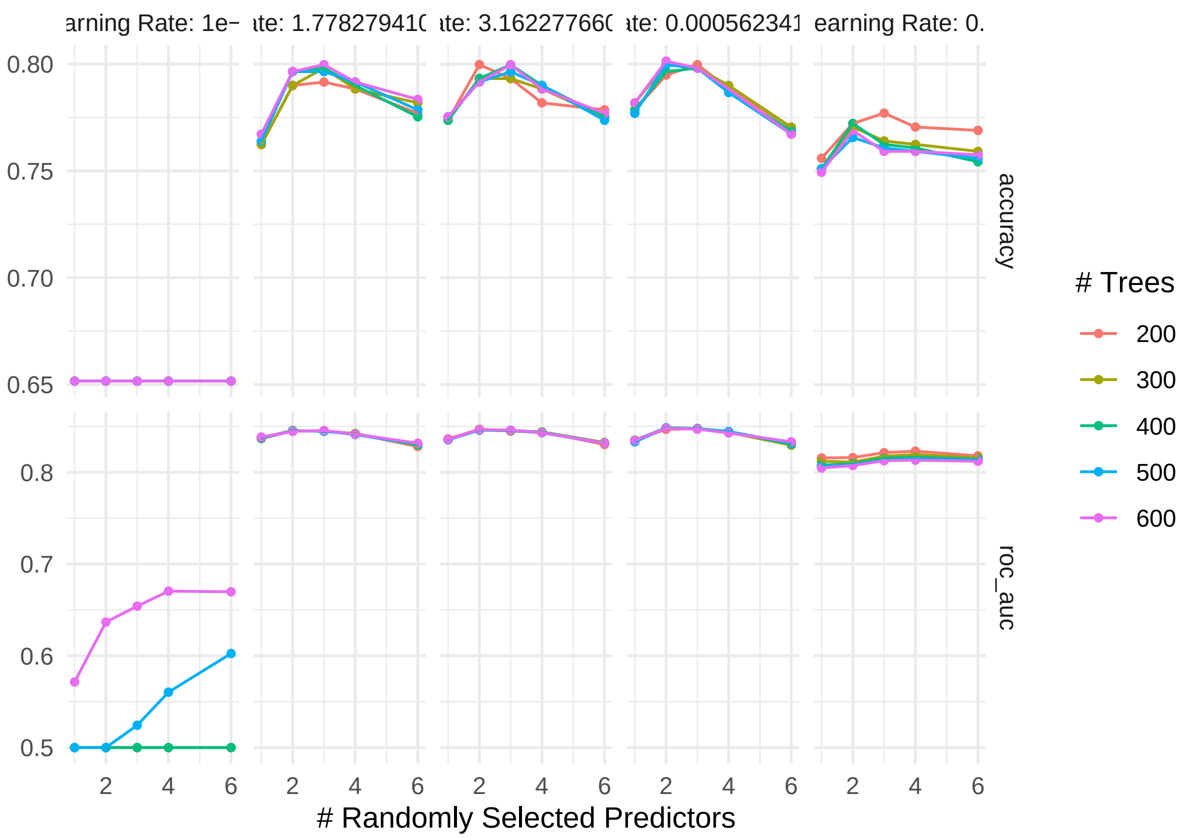
```
bt_grid <- grid_regular(mtry(range = c(1, 6)),
                        trees(range = c(200, 600)),
                        learn_rate(range = c(-10, -1)),
                        levels = 5)
bt_grid
```

```
## # A tibble: 125 x 3
##     mtry trees   learn_rate
##    <int> <int>        <dbl>
##  1     1   200 0.0000000001
##  2     2   200 0.0000000001
##  3     3   200 0.0000000001
##  4     4   200 0.0000000001
##  5     6   200 0.0000000001
##  6     1   300 0.0000000001
```

```
## 7     2   300 0.0000000001
## 8     3   300 0.0000000001
## 9     4   300 0.0000000001
## 10    6   300 0.0000000001
## # i 115 more rows
```

```r
tune_bt_class <- tune_grid(
  bt_class_wf,
  resamples = db_fold,
  grid = bt_grid
)
save(tune_bt_class, file = "tune_bt_class.rda")
```

```r
load("tune_bt_class.rda")
autoplot(tune_bt_class) + theme_minimal()
```



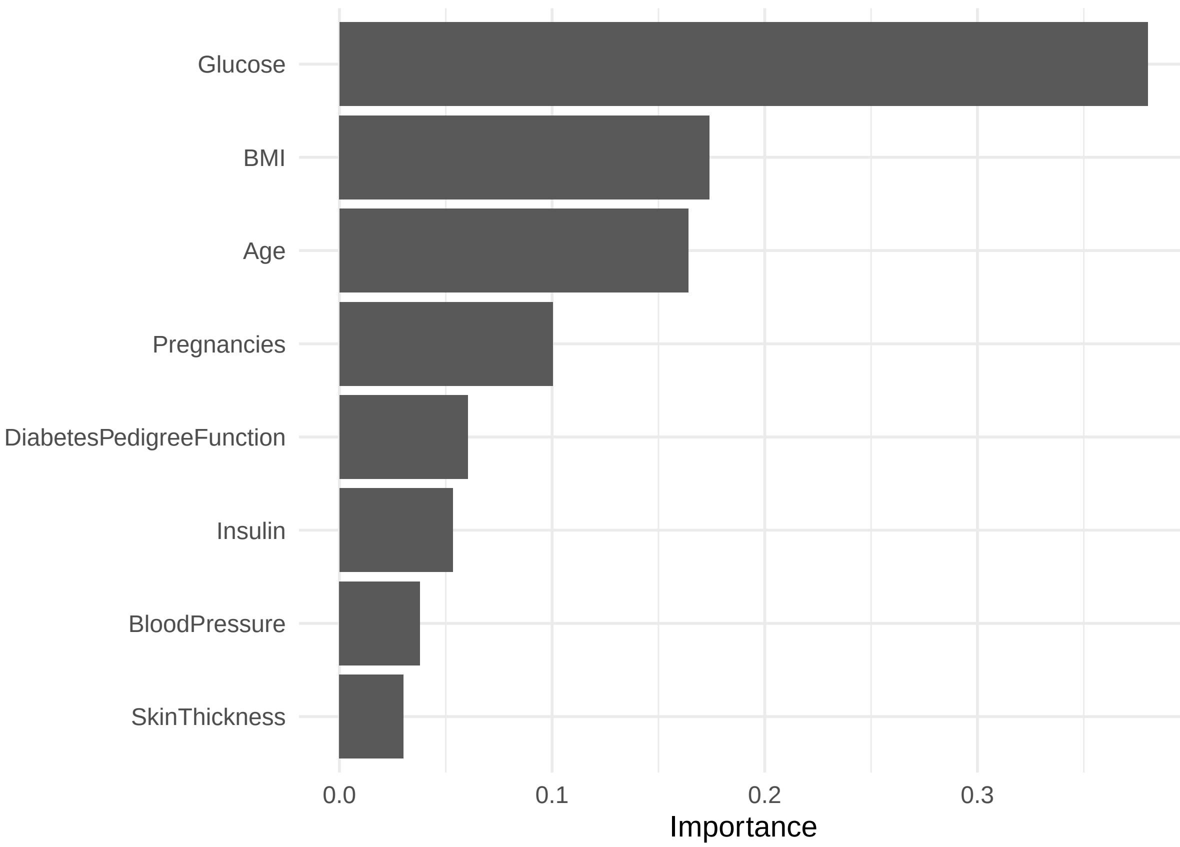

```r
show_best(tune_bt_class, n = 1)
```

```
## Warning: No value of 'metric' was given; metric 'roc_auc' will be used.

## # A tibble: 1 x 9
##    mtry trees learn_rate .metric .estimator  mean     n std_err .config
##   <int> <int>      <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     2   400   0.000562 roc_auc binary     0.849     6  0.0145 Preprocessor1_M~
```

```r
best_bt_class <- select_best(tune_bt_class)
```

```
## Warning: No value of ‘metric‘ was given; metric ’roc_auc’ will be used.
```

```r
bt_mode_fit <- finalize_workflow(bt_class_wf, best_bt_class)
bt_mode_fit <- fit(bt_mode_fit, db_train)

bt_mode_fit %>% extract_fit_parsnip() %>%
  vip() +
  theme_minimal()
```



Compute the accuracy measures and create a confusion matrix

```r
bt_predictions <- augment(bt_mode_fit, new_data = db_test)
bt_roc_curve <- roc(bt_predictions$Outcome, bt_predictions$.pred_0)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

```r
bt_auc_value <- auc(bt_roc_curve)
bt_conf_matrix <- conf_mat(bt_predictions, truth = Outcome, estimate = .pred_class)

print(paste("Boosted Trees AUC:", bt_auc_value))
```

```
## [1] "Boosted Trees AUC: 0.77962962962963"
```

```r
print("Boosted Trees Confusion Matrix:")
```

```
## [1] "Boosted Trees Confusion Matrix:"
```

```r
print(bt_conf_matrix)
```

```
##           Truth
## Prediction  0  1
##          0 83 28
##          1 17 26
```