

Decision Trees and Random Forest

Stat 220

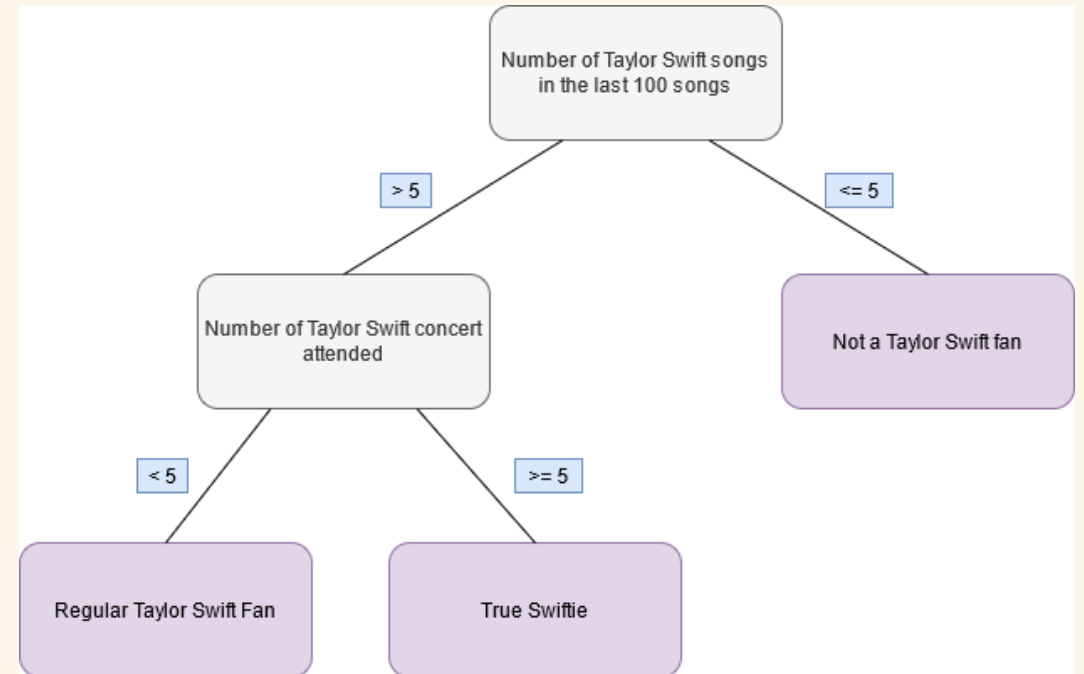
Bastola

November 11 2022

Decision Tree

- learns how to best split the dataset into smaller and smaller subsets to predict the target value

- Data is continuously split according to a certain parameter
- Two main entities:
 - **nodes**: where the data is split
 - **leaves**: decisions or final outcomes



Decision Tree

Use features to make subsets of cases that are as similar (“pure”) as possible with respect to the outcome

- Start with all observations in one group
- Find the variable/feature/split that best separates the outcome
- Divide the data into two groups (leaves) on the split (node)
- Within each split, find the best variable/split that separates the outcomes
- Continue until the groups are too small or sufficiently “pure”

Dataset

```
data(PimaIndiansDiabetes2)
db <- PimaIndiansDiabetes2 %>% drop_na() %>%
  mutate(diabetes = fct_relevel(diabetes, ref = "neg"))
```

```
glimpse(db)
Rows: 392
Columns: 9
$ pregnant <dbl> 1, 0, 3, 2, 1, 5, 0, 1, 1, 3, 11, 10, 1, 13, 3, 3, 4, 4, 3, 9...
$ glucose <dbl> 89, 137, 78, 197, 189, 166, 118, 103, 115, 126, 143, 125, 97,...
$ pressure <dbl> 66, 40, 50, 70, 60, 72, 84, 30, 70, 88, 94, 70, 66, 82, 76, 5...
$ triceps <dbl> 23, 35, 32, 45, 23, 19, 47, 38, 30, 41, 33, 26, 15, 19, 36, 1...
$ insulin <dbl> 94, 168, 88, 543, 846, 175, 230, 83, 96, 235, 146, 115, 140, ...
$ mass <dbl> 28.1, 43.1, 31.0, 30.5, 30.1, 25.8, 45.8, 43.3, 34.6, 39.3, 3...
$ pedigree <dbl> 0.167, 2.288, 0.248, 0.158, 0.398, 0.587, 0.551, 0.183, 0.529...
$ age <dbl> 21, 33, 26, 53, 59, 51, 31, 33, 32, 27, 51, 41, 22, 57, 28, 2...
$ diabetes <fct> neg, pos, pos, pos, pos, pos, pos, pos, neg, pos, neg, pos, pos, n...
```

Data preparation and pre-processing

```
set.seed(314)
db_split <- initial_split(db, prop = 0.75)
db_train <- db_split %>% training()
db_test <- db_split %>% testing()
```

```
# scaling not needed
db_recipe <- recipe(diabetes ~ ., data = db_train) %>%
  step_dummy(all_nominal(), -all_outcomes()) %>%
  prep()
```

Model Specification

- **cost_complexity:** The cost complexity parameter, the minimum improvement in the model needed at each node
- **tree_depth:** The maximum depth of a tree
- **min_n:** The minimum number of data points in a node that are required for the node to be split further.

```
tree_model <- decision_tree(cost_complexity = tune(),  
                             tree_depth = tune(),  
                             min_n = tune()) %>%  
  set_engine('rpart') %>%  
  set_mode('classification')
```

Workflow

```
# Combine the model and recipe into a workflow
tree_workflow <- workflow() %>%
  add_model(tree_model) %>%
  add_recipe(db_recipe)
```

Hyperparameter tuning

```
# Create folds for cross validation on the training data set
db_folds <- vfold_cv(db_train, v = 5, strata = diabetes)
```

```
## Create a grid of hyperparameter values to test
tree_grid <- grid_random(cost_complexity(),
                          tree_depth(),
                          min_n(),
                          size = 10)
```


View grid

```
tree_grid
# A tibble: 10 × 3
  cost_complexity tree_depth min_n
      <dbl>         <int> <int>
1  5.28e-10          7      40
2  2.99e- 6          3       7
3  1.30e- 8          1      17
4  1.94e- 2         13      19
5  1.74e- 7         11      22
6  8.11e- 7          6       3
7  2.41e- 6          4      10
8  1.01e- 5         14      40
9  2.38e- 2          4      18
10 2.68e- 6         15      39
```

Tuning Hyperparameters with `tune_grid()`

```
# Tune decision tree workflow
set.seed(314)
tree_tuning <- tree_workflow %>%
  tune_grid(resamples = db_folds,
            grid = tree_grid)
```

Best model

```
# Select best model based on accuracy
best_tree <- tree_tuning %>%
  select_best(metric = 'accuracy')
```

```
# View the best tree parameters
best_tree
# A tibble: 1 × 4
  cost_complexity tree_depth min_n .config
      <dbl>         <int> <int> <chr>
1    5.28e-10           7    40 Preprocessor1_Model01
```

Finalize workflow

```
final_tree_workflow <- tree_workflow %>%  
  finalize_workflow(best_tree)
```

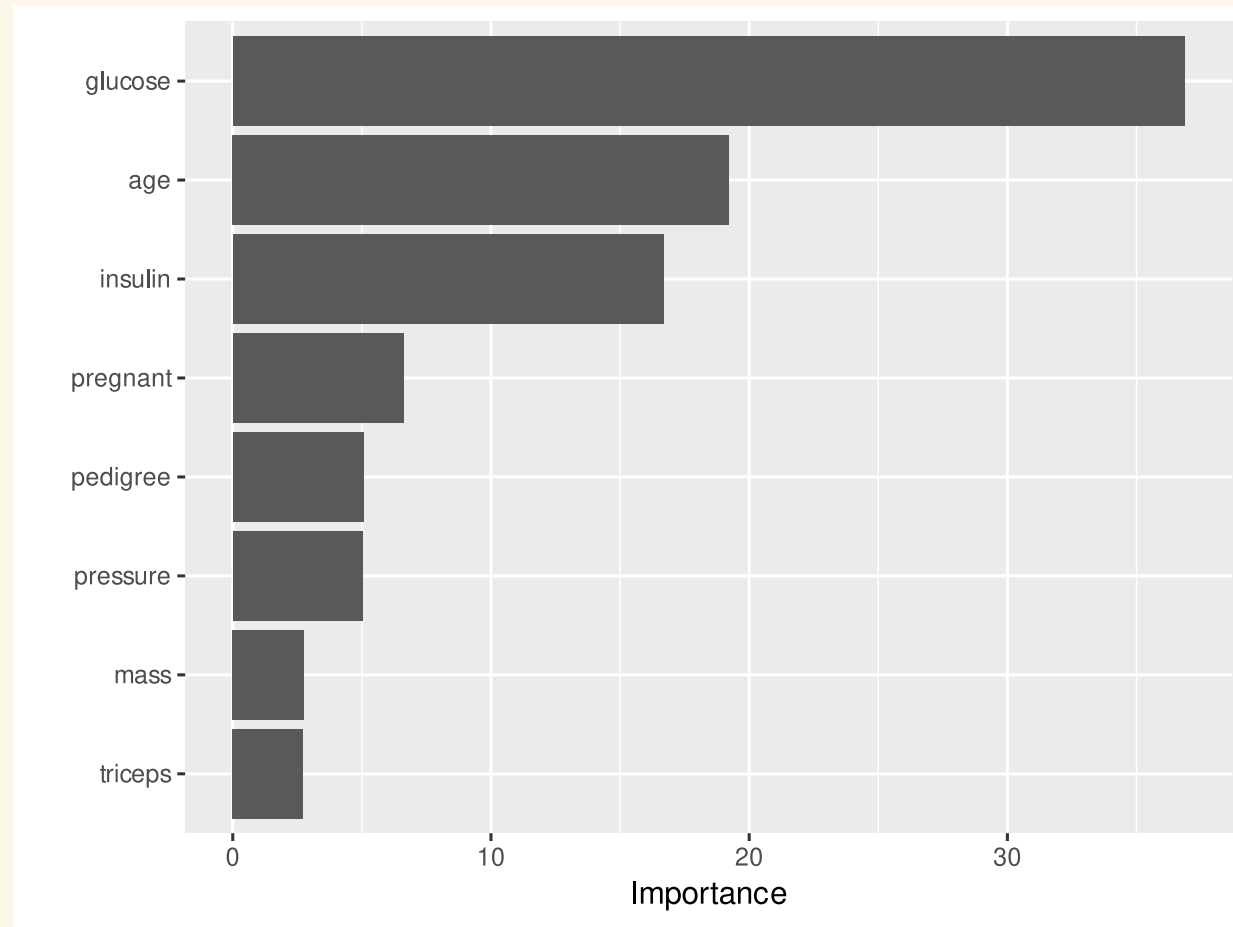
Fit the model

```
tree_wf_fit <- final_tree_workflow %>%  
  fit(data = db_train)
```

Extract fit

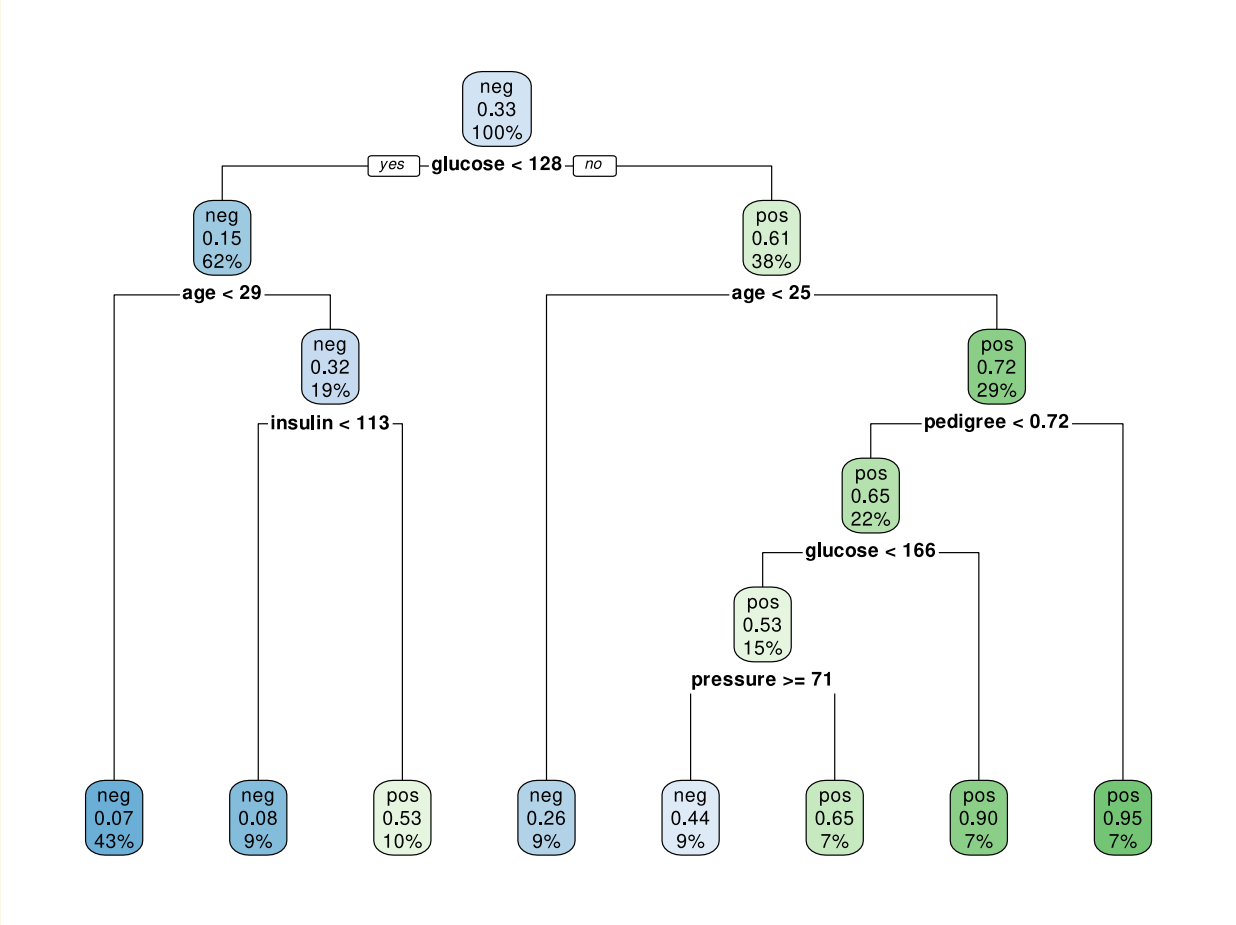
```
tree_fit <- tree_wf_fit %>%  
  extract_fit_parsnip()
```

```
vip(tree_fit)
```



Variable Importance

```
rpart.plot(tree_fit$fit, roundint = FALSE)
```



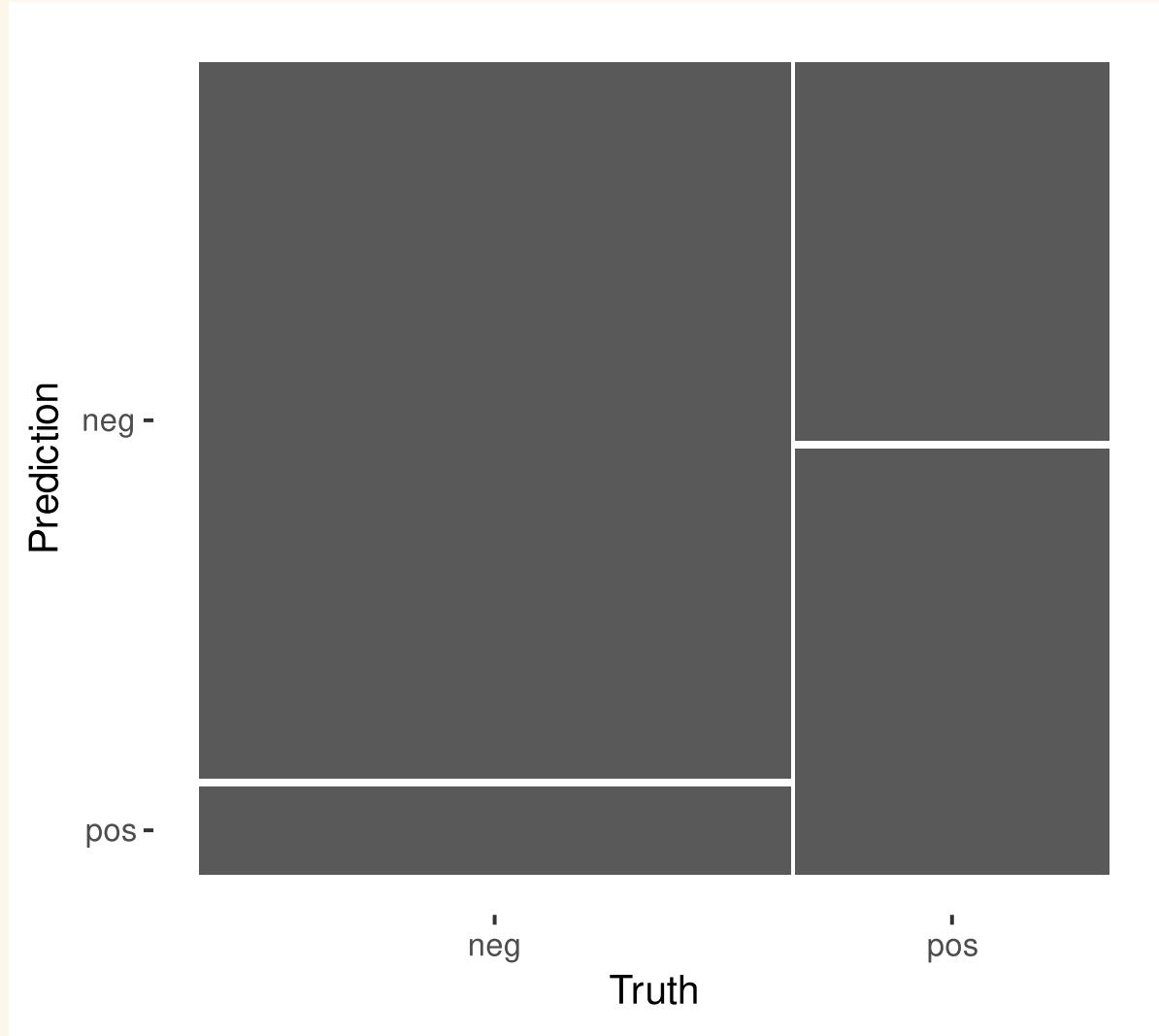
Decision Tree

Train and Evaluate With `last_fit()`

```
tree_last_fit <- final_tree_workflow %>%  
  last_fit(db_split)
```

```
tree_last_fit %>% collect_metrics()  
# A tibble: 2 × 4  
  .metric .estimator .estimate .config  
  <chr>    <chr>         <dbl> <chr>  
1 accuracy binary         0.765 Preprocessor1_Model1  
2 roc_auc  binary         0.824 Preprocessor1_Model1
```

Confusion matrix



Group Activity 1

05:00



- Get the class activity 26.Rmd file from [moodle](#)
- Let's work on group activity 1 together

Now, let's talk about random forest

Random Forest

Random forests take decision trees and construct more powerful models in terms of prediction accuracy.

- Repeated sampling (with replacement) of the training data to produce a sequence of decision tree models.
- These models are then averaged to obtain a single prediction for a given value in the predictor space.
- The random forest model selects a random subset of predictor variables for splitting the predictor space in the tree building process.

Model Specification

- **mtry:** The number of predictors that will be randomly sampled at each split when creating the tree models
- **trees:** The number of decision trees to fit and ultimately average
- **min_n:** The minimum number of data points in a node that are required for the node to be split further

Model Specification

```
rf_model <- rand_forest(mtry = tune(),  
                        trees = tune(),  
                        min_n = tune()) %>%  
  set_engine('ranger', importance = "impurity") %>%  
  set_mode('classification')
```

Workflow

```
rf_workflow <- workflow() %>%  
  add_model(rf_model) %>%  
  add_recipe(db_recipe)
```

Hyperparameter Tuning

```
## Create a grid of hyperparameter values to test
set.seed(314)
rf_grid <- grid_random(mtry() %>% range_set(c(2, 7)),
                      trees(),
                      min_n(),
                      size = 15)
```

View Grid

```
rf_grid
# A tibble: 15 × 3
  mtry trees min_n
<int> <int> <int>
1     7   609    32
2     5  1235     6
3     4  1822    29
4     5   678    16
5     4   138    14
6     3  1218    19
7     7   228    14
8     5   873     4
9     6  1387    10
10    7  1717     5
11    5   436     4
12    3  1175    16
13    6  1909    33
14    6   118     4
15    2  1003    24
```

Tuning Hyperparameters with `tune_grid()`

```
## Tune random forest workflow
set.seed(314)

rf_tuning <- rf_workflow %>%
  tune_grid(resamples = db_folds,
            grid = rf_grid)
```


Select best

```
## Select best model based on roc_auc
best_rf <- rf_tuning %>%
  select_best(metric = 'accuracy')
```

```
# View the best parameters
best_rf
# A tibble: 1 × 4
  mtry trees min_n .config
<int> <int> <int> <chr>
1      2  1003    24 Preprocessor1_Model15
```

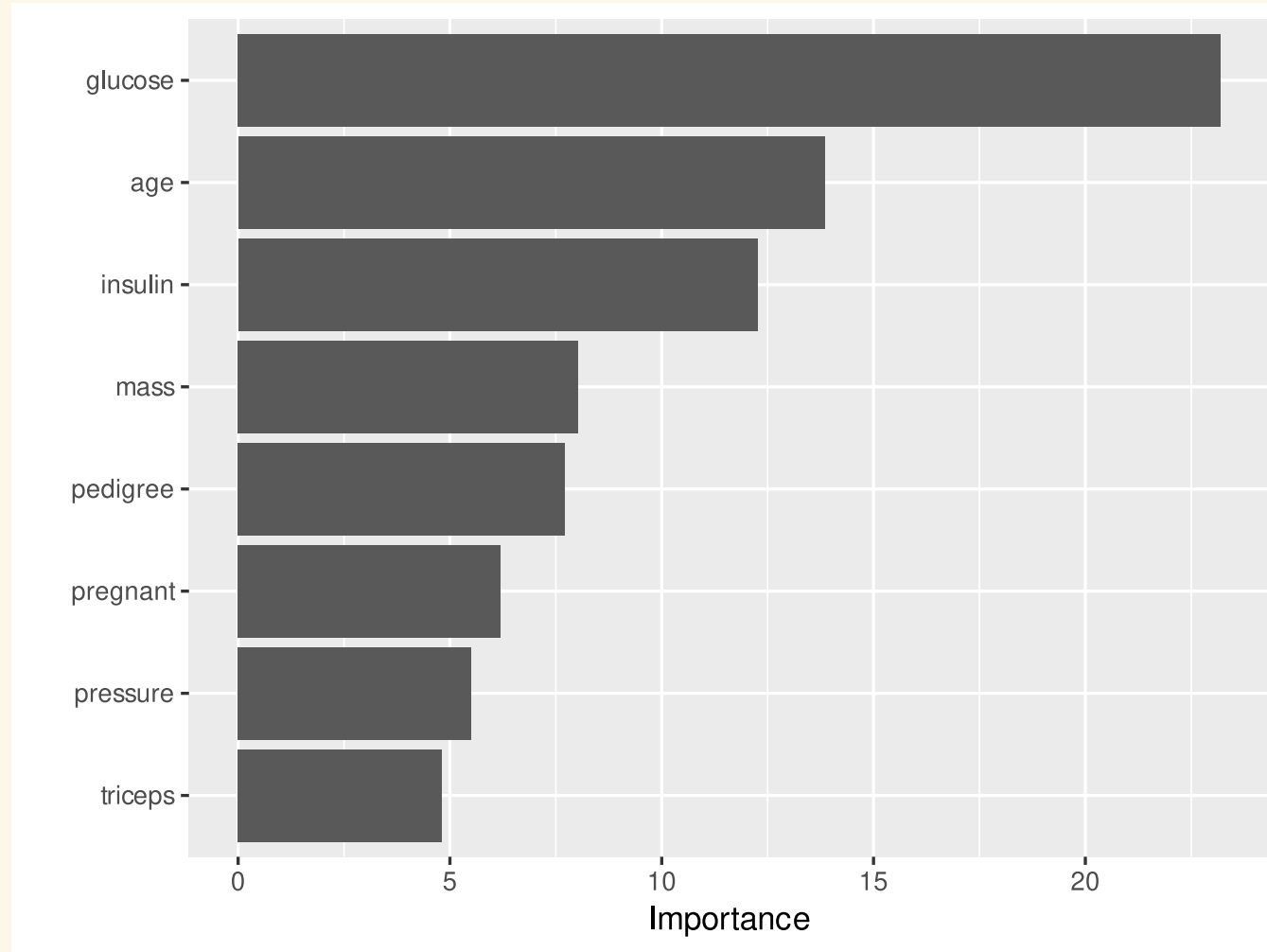
Finalize workflow

```
final_rf_workflow <- rf_workflow %>%  
  finalize_workflow(best_rf)
```

Variable Importance

```
rf_wf_fit <- final_rf_workflow %>%  
  fit(data = db_train)  
  
rf_fit <- rf_wf_fit %>%  
  extract_fit_parsnip()
```

Variable Importance



Group Activity 2

10:00



- Please continue working on group activity 2