Predicting Heart Disease

Nishith Ranjon Roy

1/26/2022

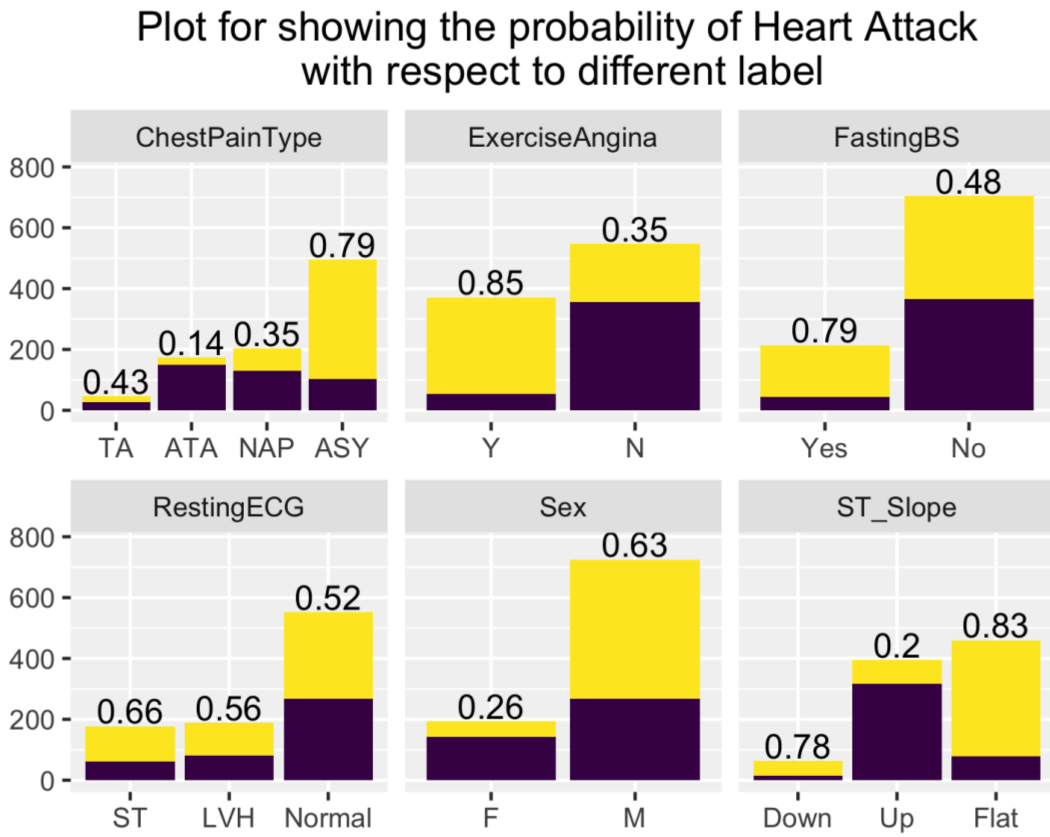
Libraries and Loading the Dataset

For this project we are going to use libaries like tidyverse and tidymodels.

Dataset

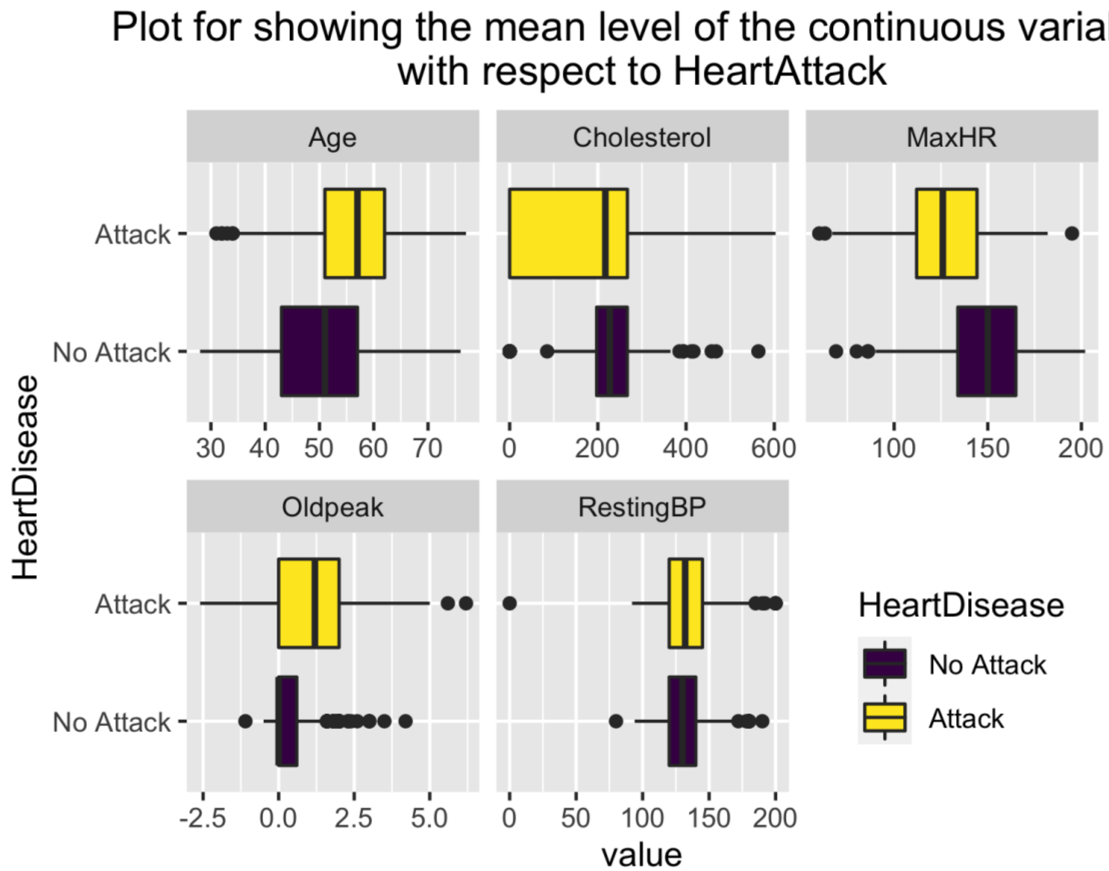
This data set contains 918 rows an 12 columns. The main goal is to predict the heart disease. This is a part of a case and control study. Here the case group (Attack) contains 508 observations on the other hand control group (No Attack) contains 410 individuals. Now we will explore and try to know the cause of the disease.

Plot for the Categorical Variables.



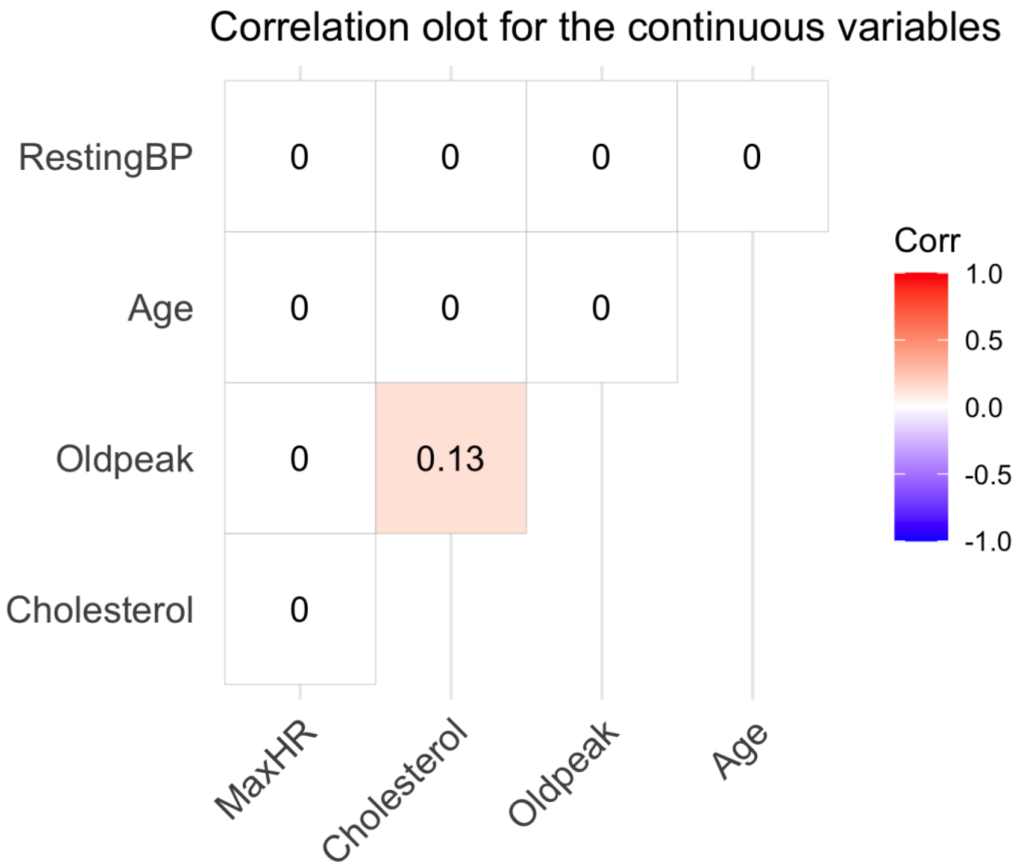
From this plot we can see that the proportion for the male population (63%) of getting the heart disease is significantly higher. For the chest pain type the "ASY" group of people has higher chance (79%) of heart disease, for the group with ExericseAngina this is 85%.

Plot for the Continuous Variables



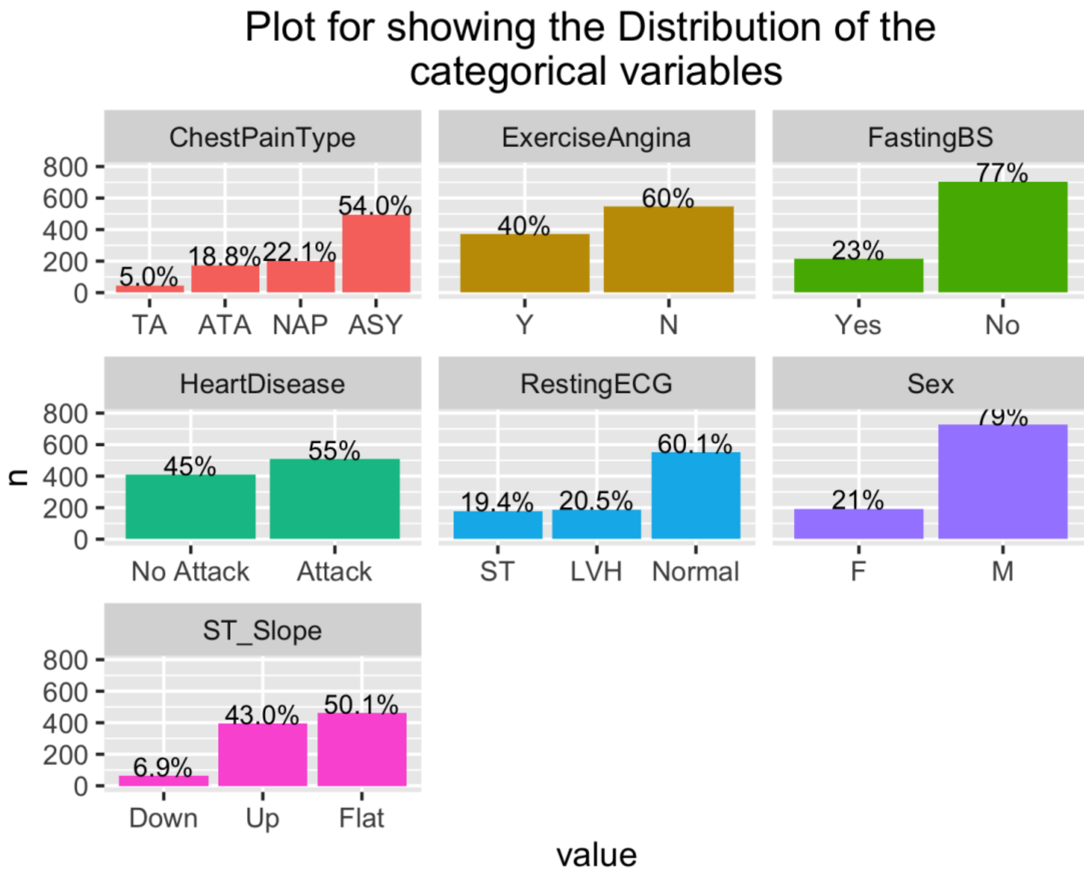
From this plot we can see that the median value of Age, Oldpeak is higher that the no attack group on the other hand median value of MaxHR is significantly low.

Correlation Plot



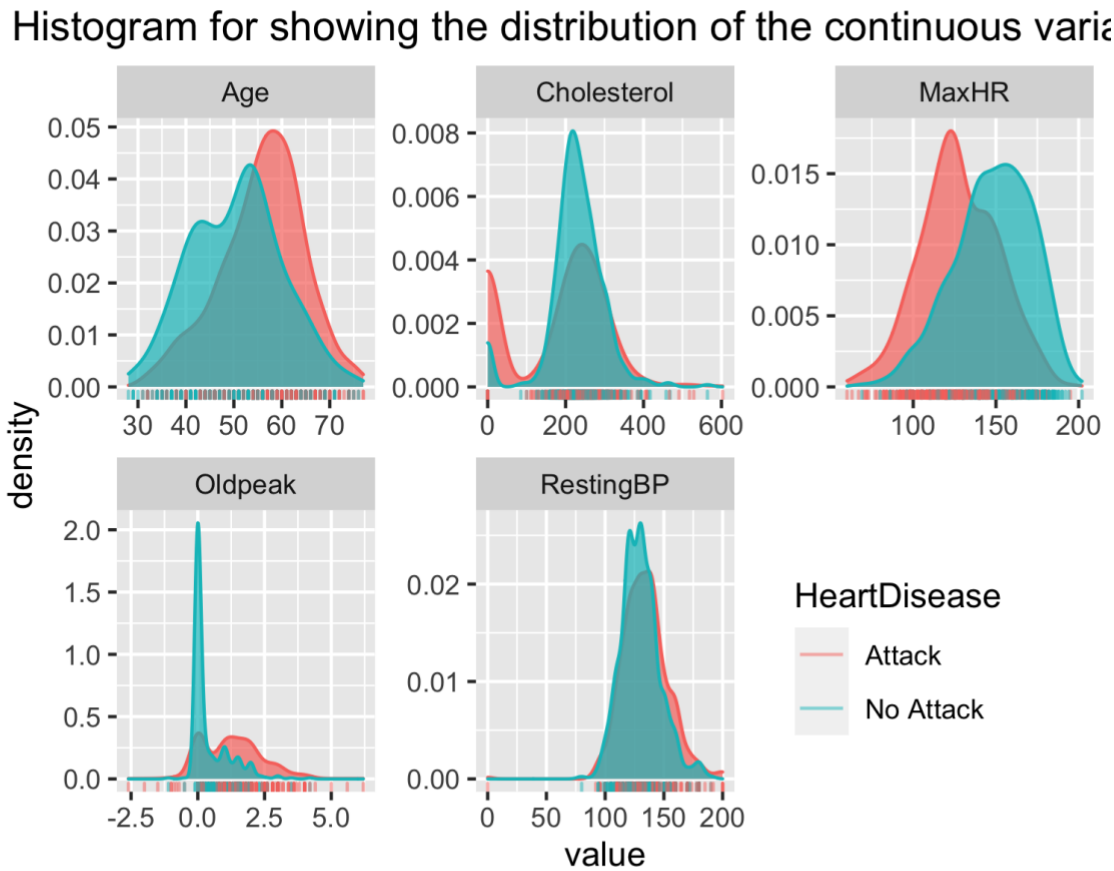
From this plot we can see that there is not a significant correlation exists among the explanatory variables. So we can move to the modeling phase.

Distribution of Categorical Variables



From this plot we can see that there is no categorical variables which has the frequency less that 5%. So np categorical variables need to be eliminated when modeling.

Distribution of the Continus Variables



From this plot we can see that there are some extreme values in the variables RestingBP and Cholestrol. But Since the values are in grater percentage so we will not eliminate the values.

Fitting the Model

Initial Split

The train set contains 779 observations and the test set contains 139 observations.

Cross Validation

We do 10-fold of 779 observations for cross validation to evaluate the performance of the model.

Evaluation Metric

Here sensitivity (The patient going to have a Heart Disease and the model also predict it as heart disease) is the main measure. And our focus would be choosing a model that gives us the highest sensitivity. This evaluation has done on 139 observations.

1. Logistic Regression

Here we will fit a logistic regression model to predict the outcome of the HeartDisease.

```
# Defining the model
model_lr <-
  logistic_reg() %>%
  set_engine("glm")

# fitting the model to the training data
fit_lr <-
  workflow() %>%
  add_model(model_lr) %>%
  add_formula(HeartDisease ~ .) %>%
  fit(juice(df_recipe))
```

Model Evaluation

.metric	.estimator	.estimate
accuracy	binary	0.8561
sensitivity	binary	0.8182
kap	binary	0.7125
roc_auc	binary	0.946

Here we have several other index for the measurement. But we will keep our focus on the sensitivity. It is still good. But it may possible to increase the performance.

2. Logistic regression

(Regularized/ L1 and L2 regularization/ Elastic net/ Ridge and Lesso Regression)

```
# Defining the model
model_enet <-
  logistic_reg(penalty = tune(),
               mixture = tune()) %>%
  set_engine("glmnet")

# HyperParameters tuning
hypParms_enet <-
  workflow() %>%
  add_model(model_enet) %>%
  add_formula(HeartDisease ~ .) %>%
  tune_grid(
    df_vfold,
    grid = 20,
    metric = metric_set(sensitivity),
    control = control_stack_grid()
  )


# fitting the model to the training data
fit_enet <-
  workflow() %>%
  add_model(model_enet) %>%
  add_formula(HeartDisease ~ .) %>%
  finalize_workflow(select_best(hypParms_enet)) %>%
  fit(juice(df_recipe))
```

Model Evaluation

.metric	.estimator	.estimate
accuracy	binary	0.8561
sensitivity	binary	0.8182
kap	binary	0.7125
roc_auc	binary	0.9453

From this model we can see that sensitivity has increased that the logistic regression model, We will fit other model and check which model is doing well.

3. Decision Tree

```r
# Defining the model
model_tree <-
  decision_tree(
    cost_complexity = tune(),
    tree_depth = tune(),
    min_n = tune()
  ) %>%
  set_mode("classification") %>%
  set_engine("rpart")

# HyperParameters tuning
hypParms_tree <-
  workflow() %>%
  add_model(model_tree) %>%
  add_formula(HeartDisease ~ .) %>%
  tune_grid(
    df_vfold,
    grid = 20,
    metric = metric_set(sensitivity),
    control = control_stack_grid()
  )


# fitting the model to the training data
fit_tree <-
  workflow() %>%
  add_model(model_tree) %>%
  add_formula(HeartDisease ~ .) %>%
  finalize_workflow(select_best(hypParms_tree)) %>%
  fit(juice(df_recipe))
```

Model Evaluation

.metric	.estimator	.estimate
accuracy	binary	0.8633
sensitivity	binary	0.8442
kap	binary	0.7256
roc_auc	binary	0.9473

The sensitivity is .8442 for this model.

4. XGboocted Trees

```
# Defining the model
model_xgb <-
  boost_tree(
    mtry = tune(),
    tree_depth = tune(),
    learn_rate = tune(),
    loss_reduction = tune()
  ) %>%
  set_mode("classification") %>%
  set_engine("xgboost")

# HyperParameters tuning
hypParms_xgb <-
  workflow() %>%
  add_model(model_xgb) %>%
  add_formula(HeartDisease ~ .) %>%
  tune_grid(
    df_vfold,
    grid = 20,
    metric = metric_set(sensitivity),
    control = control_stack_grid()
  )


# fitting the model to the training data
fit_xgb <-
  workflow() %>%
  add_model(model_xgb) %>%
  add_formula(HeartDisease ~ .) %>%
  finalize_workflow(select_best(hypParms_xgb)) %>%
  fit(juice(df_recipe))

## [22:01:27] WARNING: amalgamation/../src/learner.cc:1115: Starting in
XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
```

Model Evaluation

.metric	.estimator	.estimate
accuracy	binary	0.8777
sensitivity	binary	0.8701
kap	binary	0.7537
roc_auc	binary	0.9469

The sensitivity is 0.8701 for this model.

5. Random Forest

```
# Defining the model
model_rf <-
  rand_forest(mtry = tune(),
              trees = tune(),
              min_n = tune()) %>%
  set_mode("classification") %>%
  set_engine("ranger")

# HyperParameters tuning
hypParms_rf <-
  workflow() %>%
  add_model(model_rf) %>%
  add_formula(HeartDisease ~ .) %>%
  tune_grid(
    df_vfold,
    grid = 20,
    metric = metric_set(sensitivity),
    control = control_stack_grid()
  )


# fitting the model to the training data
fit_rf <-
  workflow() %>%
  add_model(model_rf) %>%
  add_formula(HeartDisease ~ .) %>%
  finalize_workflow(select_best(hypParms_rf)) %>%
  fit(juice(df_recipe))
```

Model Evaluation

.metric	.estimator	.estimate
accuracy	binary	0.8705
sensitivity	binary	0.8701
kap	binary	0.7388
roc_auc	binary	0.955

The sensitivity is 0.8701 for this model.

6. SVM (Support Vector Machine)

```
# Defining the model
model_svm <-
  svm_rbf(cost = tune(),
          rbf_sigma = tune(),
          margin = tune()) %>%
  set_mode("classification") %>%
  set_engine("kernlab")

# HyperParameters tuning
hypParms_svm <-
  workflow() %>%
  add_model(model_svm) %>%
  add_formula(HeartDisease ~ .) %>%
  tune_grid(
    df_vfold,
    grid = 20,
    metric = metric_set(sensitivity),
    control = control_stack_grid()
  )


# fitting the model to the training data
fit_svm <-
  workflow() %>%
  add_model(model_svm) %>%
  add_formula(HeartDisease ~ .) %>%
  finalize_workflow(select_best(hypParms_svm)) %>%
  fit(juice(df_recipe))
```

Model Evaluation

.metric	.estimator	.estimate
accuracy	binary	0.8705
sensitivity	binary	0.8571
kap	binary	0.7396
roc_auc	binary	0.9462

The sensitivity is 0.8571 for this model.

7. MARS (Multivariate Adaptive Regression Splines)

```
# Defining the model
model_mars <-
  mars(
    num_terms = tune(),
    prod_degree = tune(),
    prune_method = tune()
  ) %>%
  set_mode("classification") %>%
  set_engine("earth")

# HyperParameters tuning
hypParms_mars <-
  workflow() %>%
  add_model(model_mars) %>%
  add_formula(HeartDisease ~ .) %>%
  tune_grid(
    df_vfold,
    grid = 20,
    metric = metric_set(sensitivity),
    control = control_stack_grid()
  )


# fitting the model to the training data
fit_mars <-
  workflow() %>%
  add_model(model_mars) %>%
  add_formula(HeartDisease ~ .) %>%
  finalize_workflow(select_best(hypParms_mars)) %>%
  fit(juice(df_recipe))
```

Model Evaluation

.metric	.estimator	.estimate
accuracy	binary	0.8129
sensitivity	binary	0.7143
kap	binary	0.6319
roc_auc	binary	0.8968

The sensitivity is 0.7143 for this model.

8. KNN

```
# Defining the model
model_knn <-
  nearest_neighbor(
    neighbors = tune(),
    weight_func = tune(),
    dist_power = tune()
  ) %>%
  set_mode("classification") %>%
  set_engine("kknn")

# HyperParameters tuning
hypParms_knn <-
  workflow() %>%
  add_model(model_knn) %>%
  add_formula(HeartDisease ~ .) %>%
  tune_grid(
    df_vfold,
    grid = 20,
    metric = metric_set(sensitivity),
    control = control_stack_grid()
  )


# fitting the model to the training data
fit_knn <-
  workflow() %>%
  add_model(model_knn) %>%
  add_formula(HeartDisease ~ .) %>%
  finalize_workflow(select_best(hypParms_knn)) %>%
  fit(juice(df_recipe))
```

Model Evaluation

.metric	.estimator	.estimate
accuracy	binary	0.8633
sensitivity	binary	0.8312
kap	binary	0.7264
roc_auc	binary	0.9402

The sensitivity is 0.8312 for this model.

Stacking the models

```r
# putting together stacks
model_stacks <-
  stacks() %>%
  add_candidates(hypParms_enet) %>%
  add_candidates(hypParms_knn) %>%
  add_candidates(hypParms_mars) %>%
  add_candidates(hypParms_rf) %>%
  add_candidates(hypParms_svm) %>%
  add_candidates(hypParms_tree) %>%
  add_candidates(hypParms_xgb)

# model fitting
fit_stacks <-
  model_stacks %>%
  blend_predictions() %>%
  fit_members()
```

Model Evaluation

.metric	.estimator	.estimate
accuracy	binary	0.8993
sensitivity	binary	0.8831

Hence we can see that the accuracy and the sensitivity of the stack model is highest.