Predicting heart disease

none

12/13/2021

Class imbalance

```
df %>%
    count(HeartDisease) %>%
    pander()
```

HeartDisease	n
0	410
1	508

First of all, this is a balance dataset. We will perform the explonatory data analysis now.

```
skimr::skim(df) %>%
    select(-c(n_missing, character.min, character.max, character.empty)) %>%
    filter(skim_variable != "HeartDisease")
```

Data summary

Name	df
Number of rows	918
Number of columns	12
_____	
Column type frequency:	
character	5
numeric	6
_____	
Group variables	None

Variable type: character

skim_variable	complete_rate	n_unique	whitespace
Sex	1	2	0
ChestPainType	1	4	0
RestingECG	1	3	0
ExerciseAngina	1	2	0
ST_Slope	1	3	0

Variable type: numeric

skim_variable	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Age	1	53.51	9.43	28.0	47.00	54.0	60.0	77.0	▁▅▆▁
RestingBP	1	132.40	18.51	0.0	120.00	130.0	140.0	200.0	▁▁▃▇▁
Cholesterol	1	198.80	109.38	0.0	173.25	223.0	267.0	603.0	▃▆▇▁▁
FastingBS	1	0.23	0.42	0.0	0.00	0.0	0.0	1.0	▇▁▁▁▂
MaxHR	1	136.81	25.46	60.0	120.00	138.0	156.0	202.0	▁▃▅▆▃
Oldpeak	1	0.89	1.07	-2.6	0.00	0.6	1.5	6.2	▁▅▆▁▁

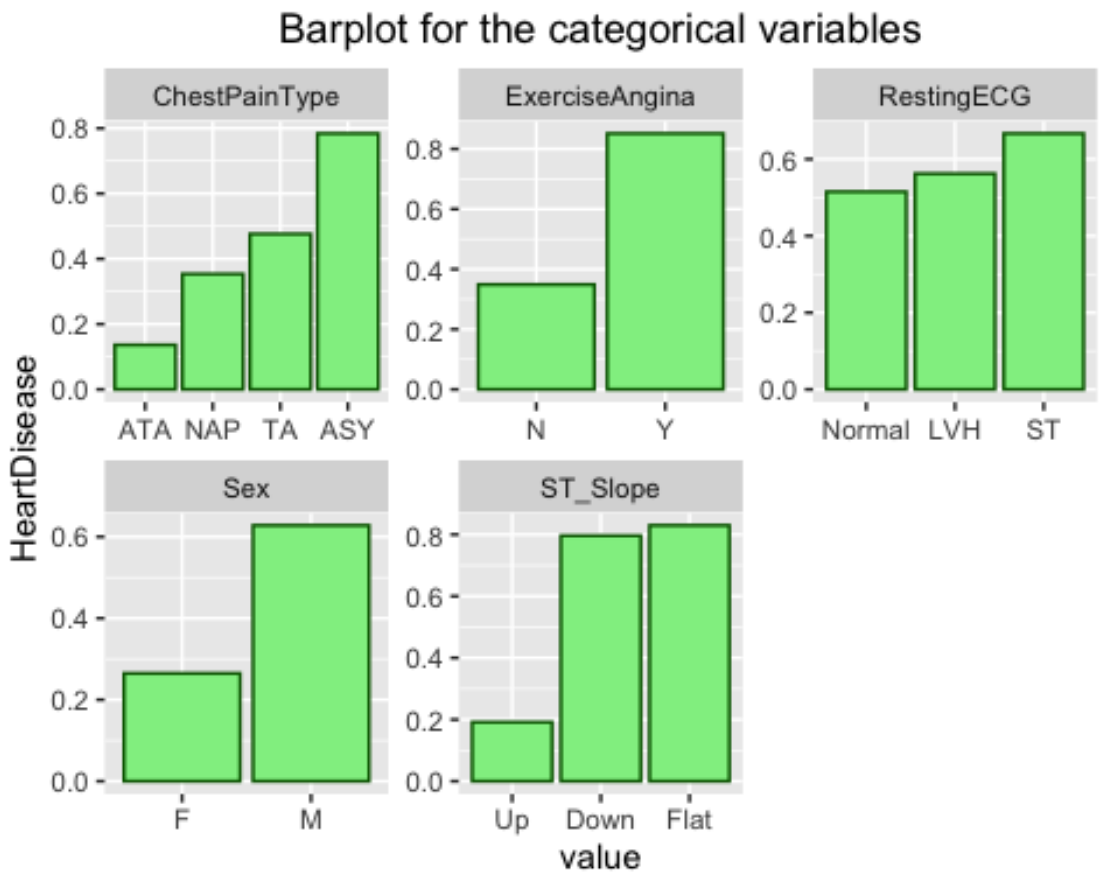
This dataset total contains 918 rows and 12 columns. Among those 12 columns 1 is outcome variable and 11 is predictor. 5 of those predictors are character and 6 of those predictors are numeric. The source of the data is {**Kaggle**} [https://www.kaggle.com/fedesoriano/heart-failure-prediction].

Train test split

```
df_split <-
    df %>%
    initial_split(prop = .9 ,strata = "HeartDisease")

df <- training(df_split)
```

Boxplot for the categorical variables

```
df %>%
    select(is.character, HeartDisease) %>%
    pivot_longer(-HeartDisease) %>%
    group_by(name, value) %>%
    summarise_all(mean) %>%
    ungroup() %>%
    mutate(value = tidytext::reorder_within(value,by = HeartDisease, within =
name)) %>%
    ggplot(aes(value, HeartDisease)) +
    geom_col(fill = "lightgreen", col = "darkgreen") +
    tidytext::scale_x_reordered() +
    facet_wrap(~name, scales = "free") +
    labs(title = "Barplot for the categorical variables")
```



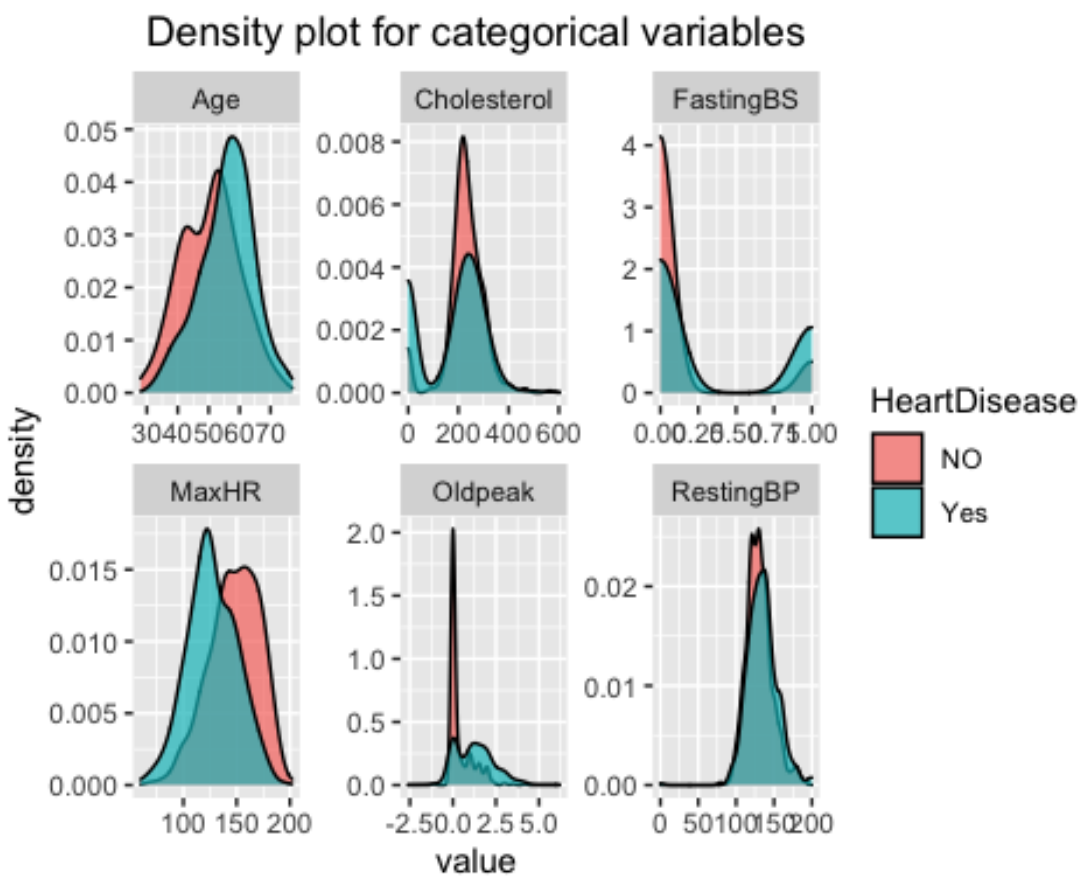
So, we can see that for variable

```
- ExerciseAngina level (N) has lower chance of heart disease.
- ChestPain Type level (ATA, NAP, TA) has lower chance of heart disease.
- SEX            level (F) has lower chance of heart disease.
- ST_Slope       Level (Up) has lower chance of heart disease.
```

So those categorical variables could be good predictor for the heart disease.

Density plot for the numeric variables

```
df %>%
    select(is.numeric) %>%
    pivot_longer(-HeartDisease) %>%
    mutate(HeartDisease = factor(HeartDisease, 0:1, c("NO", "Yes"))) %>%
    ggplot(aes(value, fill = HeartDisease)) +
    geom_density(alpha = .7) +
    facet_wrap(~name, scales = "free") +
    labs(title = "Density plot for categorical variables")
```



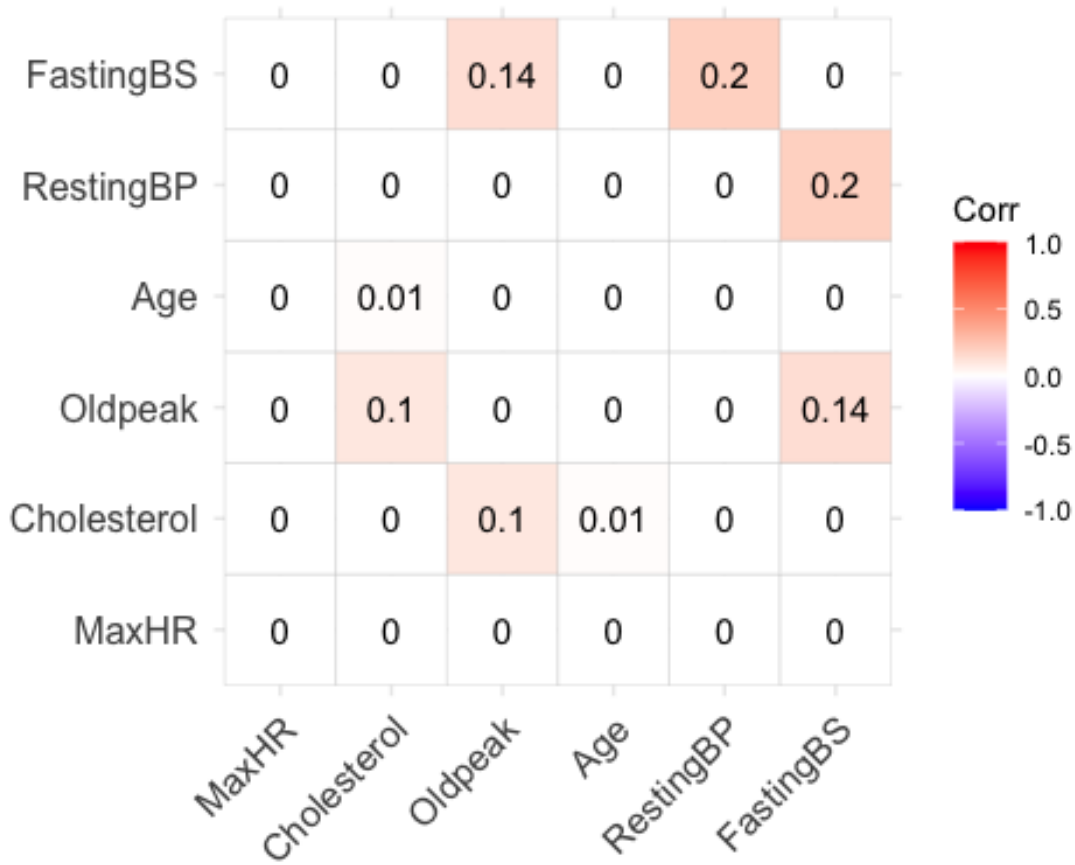
From this plot we can see that the variables like

```
- Age
- MaxHR
- Oldpeak
```

may be some good predictor canditate.

Correlation among the numeric predictor

```
df %>%
    select(is.numeric, -HeartDisease) %>%
    ggcorrplot::cor_pmat() %>%
    ggcorrplot::ggcorrplot(hc.order = T, lab = T, p.mat = )
```



There is not much correlation exists among the predictor variables.

Modeling

Defining the models

```
models <- list()
models$lr_specs <-
    logistic_reg(mode = "classification") %>%
    set_engine("glm")

models$spline_specs <-
    mars(
        mode = "classification",
        num_terms = tune(),
        prod_degree = tune(),
        prune_method = tune()
    ) %>%
    set_engine("earth")

models$rf_specs <-
    rand_forest(
        mode = "classification",
        trees = 1000,
        min_n = tune()
    ) %>%
    set_engine("randomForest")

models$bt_specs <-
    boost_tree(
        mode = "classification",
        trees = 1000,
        min_n = tune(),
        tree_depth = tune(),
        learn_rate = tune()
    ) %>%
    set_engine("xgboost")

models$svm_specs <-
    svm_rbf(
        mode = "classification",
        cost = tune(),
        rbf_sigma = tune(),
        margin = tune()
    ) %>%
    set_engine("kernlab")
```

Cross validation and recipe

```
df_recipe <-
    df %>%
    recipe(HeartDisease ~ .) %>%
    step_string2factor(all_nominal()) %>%
    step_mutate(HeartDisease = factor(HeartDisease, 0:1, c("No", "Yes"))) %>%
    step_normalize(all_numeric(), - all_outcomes()) %>%
    prep()

set.seed(123)
df_cv <-
    df_recipe %>%
    juice() %>%
    vfold_cv(6)
```

Function for model fitting

```
model_fit <-
  function(x) {
    set.seed(123)

    if (nrow(parameters(x)) != 0) {
      vlu_fw <-
        workflow() %>%
        add_model(x) %>%
        add_formula(formula(df_recipe))


      print("hyperParameter Training")
      x <-
        vlu_fw %>%
        tune_grid(
          resamples = df_cv,
          grid = grid_latin_hypercube(parameters(x), size = 25),
          metrics = metric_set(roc_auc)
        )
      print("Model Training")
      x <-
        vlu_fw %>%
        finalize_workflow(select_best(x)) %>%
        fit(juice(df_recipe))
    } else {
      print("Model Training")
      x <-
        workflow() %>%
        add_model(x) %>%
        add_formula(formula(df_recipe)) %>%
        fit(juice(df_recipe))
    }
```

```
    return(x)
  }
```

Model fitting

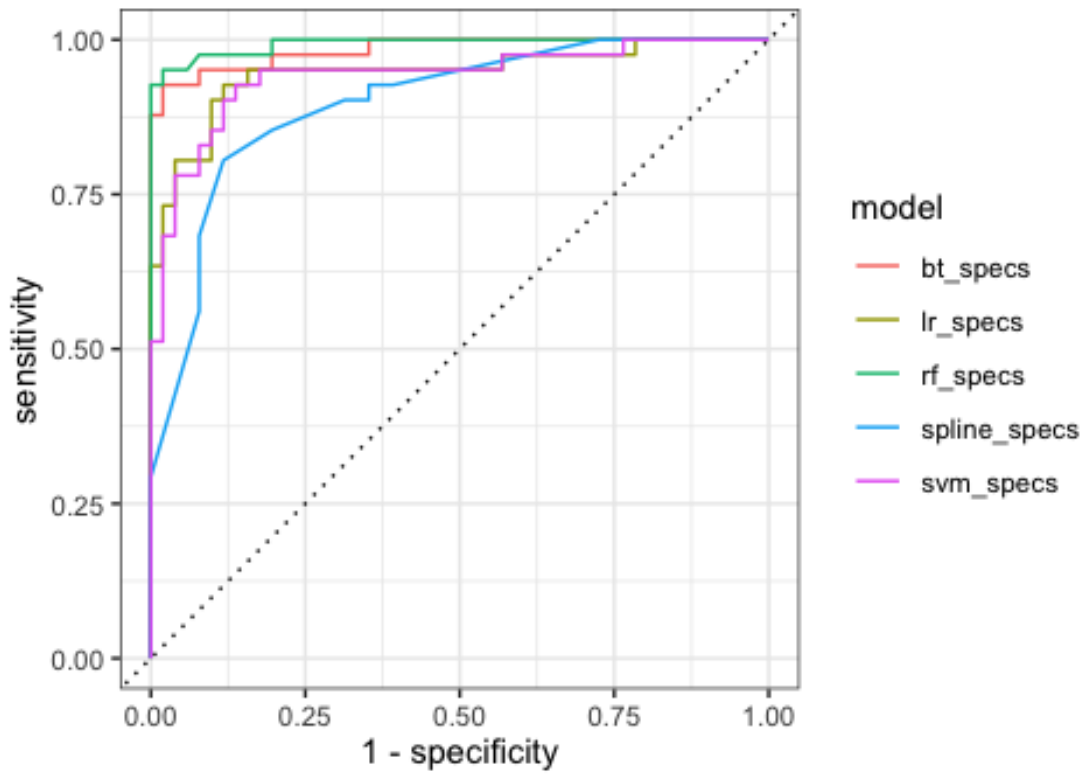
```
# fit_models <- list()
# for (i in 1:length(models)) {
#     fit_models[[i]] <- model_fit(models[[i]])
# }

fit_models <- readRDS("fitted.Rds")
```

Predicting the test dataset

```
x <-
tibble(model = names(models), fit_models) %>%
    mutate(
        tibble = map(fit_models,
                     ~ predict(.x, bake( df_recipe, testing(df_split) ), type
= "prob")),
        test = list(
            testing(df_split) %>%
                bake(object = df_recipe) %>%
                pull(HeartDisease)
                        )
        ) %>%
    select(-fit_models) %>%
    unnest()
```

Roc AUC curve

```
x %>%
    group_by(model) %>%
    roc_curve(test, .pred_No) %>%
    autoplot()
```



From this graph it is not clear that which model is performing better. SO we will find the value of area under the curve.

Area under curve

```
x %>%
    group_by(model) %>%
    roc_auc(test, .pred_No) %>%
    arrange(-.estimate) %>%
    pander()
```

model	.metric	.estimator	.estimate
rf_specs	roc_auc	binary	0.9931
bt_specs	roc_auc	binary	0.9837
lr_specs	roc_auc	binary	0.946
svm_specs	roc_auc	binary	0.9407
spline_specs	roc_auc	binary	0.8984

So, the models svm, logistic regression and random forest provide almost equal performance. So we will choose any of those 3 as our final model.

Accuracy

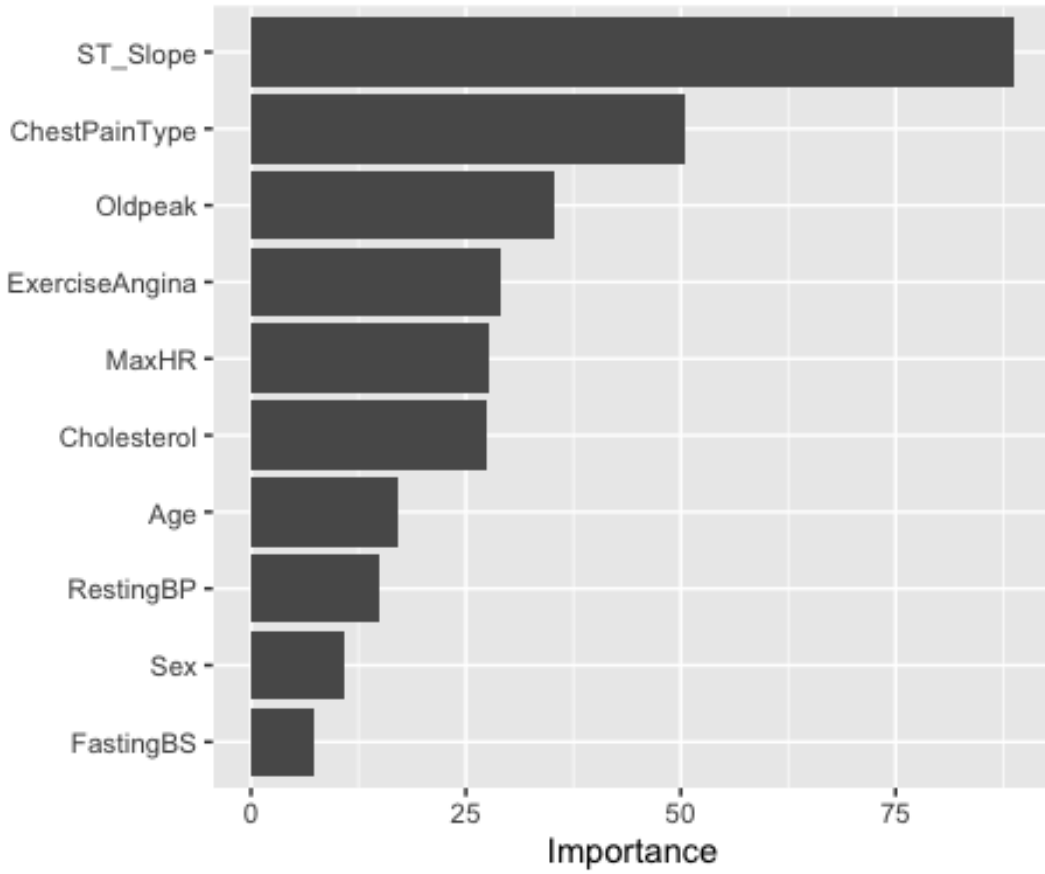
```
x %>%
    group_by(model) %>%
    mutate(.pred_Yes = ifelse(.pred_Yes > .5, "Yes", "No"),
           .pred_Yes = factor(.pred_Yes,c("No", "Yes"))) %>%
    accuracy(test, .pred_Yes) %>%
    arrange(-.estimate) %>%
    pander()
```

model	.metric	.estimator	.estimate
rf_specs	accuracy	binary	0.9565
bt_specs	accuracy	binary	0.9348
lr_specs	accuracy	binary	0.9022
svm_specs	accuracy	binary	0.8913
spline_specs	accuracy	binary	0.837

From here we can choose Ranndomforest for our desiarable model since it has the highest accuracy.

Variable importance

```
fit_models[[3]] %>%
    pull_workflow_fit() %>%
    vip::vip()
```



According to this variance importance graph, varaibles “ST_Slope”, “ChestPainType” and “Oldpeak” are the most 3 important variables.

Confusion matrix of the test set

```
x %>%
    filter(model == "rf_specs") %>%
    mutate(.pred_Yes = ifelse(.pred_Yes > .5, "Yes", "No"),
           .pred_Yes = factor(.pred_Yes,c("No", "Yes"))) %>%
    select(Pred = .pred_Yes, true = test)  %>%
    table()

##      true
## Pred  No Yes
##   No  39   2
##   Yes  2  49
```