Final Project

Shuhei Kaneko

2022-11-11

Introduction

Today, diabetes is one of the major chronic diseases in the world. Center of Disease Control and Prevention (CDC) estimates that 37.3 million Americans (approximately 10 percent of total population) have diabetes; however, about 1 in 5 people with diabetes do not recognize they actually have it. One reason of their unawareness is that there are few subjective symptoms in the early stages of diabetes. Also, because of this traits of diabetes, it is not uncommon for the disease to be discovered only after it has progressed.

The purpose of this project is to predict the history of diabetes by using limited set of information. If the risk of developing diabetes can be accurately predicted based on information that is easily available through administrative records or simple questionnaires (such as demographic information, height, and weight), it would be possible to encourage only high-risk population to undergo physical examination, which could improve the national healthcare budget and population health at the same time.

Data Set

I use 2021 (latest) version of Behavioral Risk Factor Surveillance System (BRFSS) dataset, which is the nation's representative telephone surveys that collect state data about their health-related risk behaviors, chronic health conditions, and use of preventive services. BRFSS collects data from all 50 states as well as the District of Columbia and three U.S. territories. Starting from 1980, the rough sample size is over 400,000 per year, which I believe is a sufficient number to predict some phenomena using the machine learning method. Although BRFSS contains an abundant number of variables especially on health habit (e.g. smoking, drinking, fruits intake, etc.), I will not utilize them. There are two main reasons. First, we have to keep in mind that these variables can be significantly affected by the diagnosis of diabetes. Hence, it might not be appropriate to predict the history of diabetes diagnosis, which makes me use the set of variables which is less likely to be affected by the diagnosis. Second, as the main objective of this study is to classify the diabetes patients as precisely as possible using a limited information that is not costly to gather. In this sense, I believe that the exclusion of health behavior variables is reasonable. In the next section, I briefly explain about the choice of predictor and cleaning procedure.

Preliminary steps

Data Cleaning / List of Variables Data cleaning is executed by the R script named `data_processing.R`. Please refer to this file for the detailed steps. Here, I give a succinct explanation about the steps I took.

1. Import the raw data. Because the format of raw data is `.xpt` (SAS XPORT File), I used `foreign::read.xport` package.
2. Select and re-code the variables. If the answer is 9 or 99, it means missing. Hence, I replaced them with `NA`. Also, I replace 7 or 77 (refuse to answer) with `NA`.
3. Drop the sample in Puerto Rico (Focusing just on 50 states + DC).
4. As `age` variable is truncated at 80 (coded as 80 if older than 80), I dropped these truncated sample in order to avoid the measurement error in predictor.
5. Save the processed data as `brfss_final.csv`.

List of all the variables used in this project is described in `BRFSS_final_Codebook.pdf` file.

- `state`: State FIPS Code (Including Washington DC, excluding Guam, Puerto Rico, and Virgin Islands)
- `sex`: Sex of Respondents
- `metro_status`: Metropolitan Status
- `urban_status`: Urban/Rural Status
- `race`: Race of Respondents
- `marital`: Marital Status
- `education`: Education Level
- `home_rent_own`: Own or Rent Home
- `veteran`: Veteran Status
- `employ`: Employment Status
- `num_children`: Number of Total Children Younger than 18 Years Old (continuous)
- `income_gr`: Income Level
- `age`: Age of Respondents (continuous)
- `ins_dummy`: Dummy for Having Any Health Insurance
- `height_meter`: Height in meter (continuous)
- `weight_kg`: Weight in kilogram (continuous)
- `bmi`: Body Mass Index (BMI) (continuous)
- `diabete`: History of Diabetes Diagnosis

Loading packages First, I load the necessary packages using library function.

```
library(tidyverse)
library(tidymodels)
library(corrplot)
library(ggplot2)
library(MASS)
library(discrim)
library(ranger) #random forest
library(xgboost)
```

Import csv file (processed using `data_processing.R` script) as a tibble:

```
set.seed(1234)
brfss <- read_csv("cleaned_data/brfss_final.csv")

# Drop sample missing in any variables.
brfss <- brfss %>%
  drop_na()

brfss <- brfss %>% sample_n(size=10000)
```

Here, I dropped observations if any of variables are missing. Also, to reduce the computation time, I decided to randomly pick up 10000 samples. #### Data split

Split the data into training and testing set with stratifying by outcome.

```
set.seed(12)
diabete_split <- initial_split(brfss,
                               prop = 0.75,
                               strata = diabete)
diabete_train <- training(diabete_split)
diabete_test <- testing(diabete_split)

#Make sure the dimension of dataset is correct.
dim(diabete_train)
```

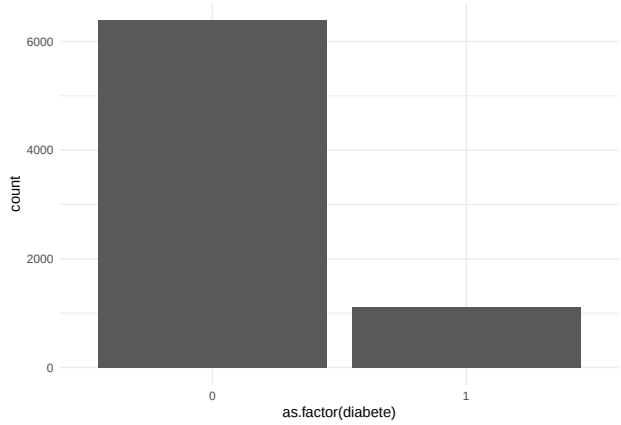
```
## [1] 7500   21
```

```
dim(diabete_test)
```

```
## [1] 2500   21
```

Descriptive statistics The distribution of outcome variables by ggplot. We can see that the fraction of people with diabetes history is approximately 14.8 percent, which is slightly higher than the estimate by CDC, but remember that I coded Yes for pre-diabete or borderline people.

```
hist_diabete <- diabete_train %>%
  ggplot(aes(as.factor(diabete))) +
  geom_bar() +
  theme_minimal()
plot(hist_diabete)
```



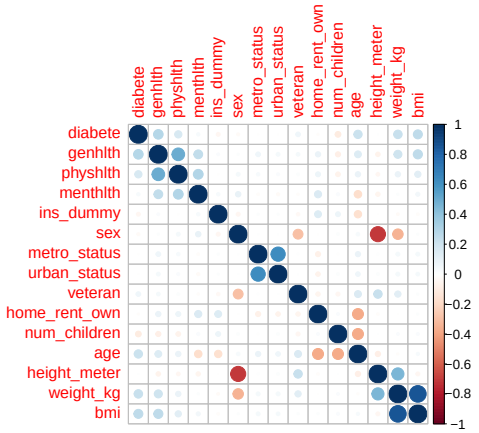

```
mean_outcome <- diabete_train %>%
  summarise(frac_diabete = mean(as.numeric(diabete)))

mean_outcome
```

```
## # A tibble: 1 x 1
##   frac_diabete
##          <dbl>
## 1        0.148
```

Next, I check the correlation between outcome (i.e. diabetes) and predictors: I can see that general health description, age weight, and BMI may be a good predictor because the correlation seems to be more significant than others.

```
diabete_train %>%
  dplyr::select(diabete,genhlth,physhlth,
         menthlth, ins_dummy,sex,
         metro_status,urban_status,
         veteran,home_rent_own,
         num_children,age, height_meter,
         weight_kg, bmi) %>%
  cor(use = "complete.obs") %>%
  corrplot(method = "circle")
```



For the convenience in the later analysis, I change the format of categorical variabls from numeric to factor:

```
# Define categorical variables as factor
## Training data
diabete_train <- diabete_train %>%
  mutate(state = as.factor(state)) %>%
  mutate(genhlth = as.factor(genhlth)) %>%
  mutate(sex = as.factor(sex)) %>%
  mutate(metro_status = as.factor(metro_status)) %>%
  mutate(urban_status = as.factor(urban_status)) %>%
  mutate(race = as.factor(race)) %>%
  mutate(marital = as.factor(marital)) %>%
  mutate(education = as.factor(education)) %>%
  mutate(home_rent_own = as.factor(home_rent_own)) %>%
  mutate(veteran = as.factor(veteran)) %>%
  mutate(employ = as.factor(employ)) %>%
  mutate(income_gr = as.factor(income_gr)) %>%
  mutate(ins_dummy = as.factor(ins_dummy)) %>%
  mutate(diabete = as.factor(diabete))

## Test data
diabete_test <- diabete_test %>%
  mutate(state = as.factor(state)) %>%
  mutate(genhlth = as.factor(genhlth)) %>%
  mutate(sex = as.factor(sex)) %>%
  mutate(metro_status = as.factor(metro_status)) %>%
  mutate(urban_status = as.factor(urban_status)) %>%
  mutate(race = as.factor(race)) %>%
  mutate(marital = as.factor(marital)) %>%
  mutate(education = as.factor(education)) %>%
  mutate(home_rent_own = as.factor(home_rent_own)) %>%
  mutate(veteran = as.factor(veteran)) %>%
  mutate(employ = as.factor(employ)) %>%
  mutate(income_gr = as.factor(income_gr)) %>%
  mutate(ins_dummy = as.factor(ins_dummy)) %>%
  mutate(diabete = as.factor(diabete))
```

I apply 5-folds cross validation to avoid the possibility of overfitting to the training dataset:

```
diabete_folds <- vfold_cv(diabete_train, v = 5, strata = diabete)
```

Recipe I convert all categorical predictors to dummy variables. Also, I scale and ceter all the predictors.

```
diabete_recipe<-recipe(diabete ~ .,                          data = diabete_train) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_scale(all_predictors()) %>%
  step_center(all_predictors())
```

Build models

In this project, I try the following four models:

- Logistic regression
- LDA
- Random Forest
- Boosted Trees
-

Logistic regression First model I am going to apply is simple Logistic regression. Here, I do not use regularization method. To assess the prediction performance for training set, I use five-folds cross validation.

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

# Workflow
log_wkflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(diabete_recipe)
```

```
# tuning
tune_log <- tune_grid(object = log_wkflow, resamples = diabete_folds)

save(tune_log, file = "model_fit/logistic_tune.rda")
```

```
## LDA
# Engine
lda <- discrim_linear() %>%
  set_engine("MASS") %>%
  set_mode("classification")
# Workflow
lda_wkflow <- workflow() %>%
  add_model(lda) %>%
  add_recipe(diabete_recipe)
```

```
# Tuning
tune_lda <- tune_grid(object = lda_wkflow, resamples = diabete_folds)

save(tune_lda, file = "model_fit/lda_tune.rda")
```

Linear discriminant analysis (LDA)

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

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

Random forest Next, I set up the grids of tuning parameters. Note that because the total number of predictors is 20, `mtry` should be 20 at most.

```
rf_params <- parameters(rf_model) %>%
  update(mtry = mtry(range= c(2, 20)),
         trees = trees(range=c(1,1000)),
         min_n = min_n(range=c(1,40)))

# Define grid
rf_grid <- grid_regular(rf_params, levels = 4)
```

Then, I execute random forest model and search the optimal tuning parameter. As this procedure takes tons of times, I set `eval = FALSE` so that I do not need run again in knitting the report.

```
rf_tune <- rf_workflow %>%
  tune_grid(resamples = diabete_folds,
    grid = rf_grid)

save(rf_tune, rf_workflow, file = "model_fit/randomforest_tune.rda")
```

Boosted trees First, I set up the models, workflows, and grids in the similar way as I did for random forest model.

```
bt_model <- boost_tree(
  mode = "classification",
  min_n = tune(),
  mtry = tune(),
  learn_rate = tune(),
  trees = tune()) %>%
  set_engine("xgboost")

bt_workflow <- workflow() %>%
  add_model(bt_model) %>%
  add_recipe(diabete_recipe)

bt_params <- parameters(bt_model) %>%
  update(mtry = mtry(range= c(2, 20)),
         learn_rate = learn_rate(range = c(-5, 0.2)),
         min_n = min_n(range = c(1,40)),
         trees = trees(range = c(1,1000)))
```

```
# define grid
bt_grid <- grid_regular(bt_params, levels = 4)
```

```
bt_tune <- bt_workflow %>%
  tune_grid(
    resamples = diabete_folds,
    grid = bt_grid
    )

# save results
save(bt_tune, bt_workflow, file = "model_fit/bt_tune.rda")
```

Support vector machine

Model Analysis

- Logistic regression

```
load(file = "model_fit/logistic_tune.rda")
collect_metrics(tune_log)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.854     5 0.00263 Preprocessor1_Model1
## 2 roc_auc  binary     0.799     5 0.0107  Preprocessor1_Model1
```

- LDA

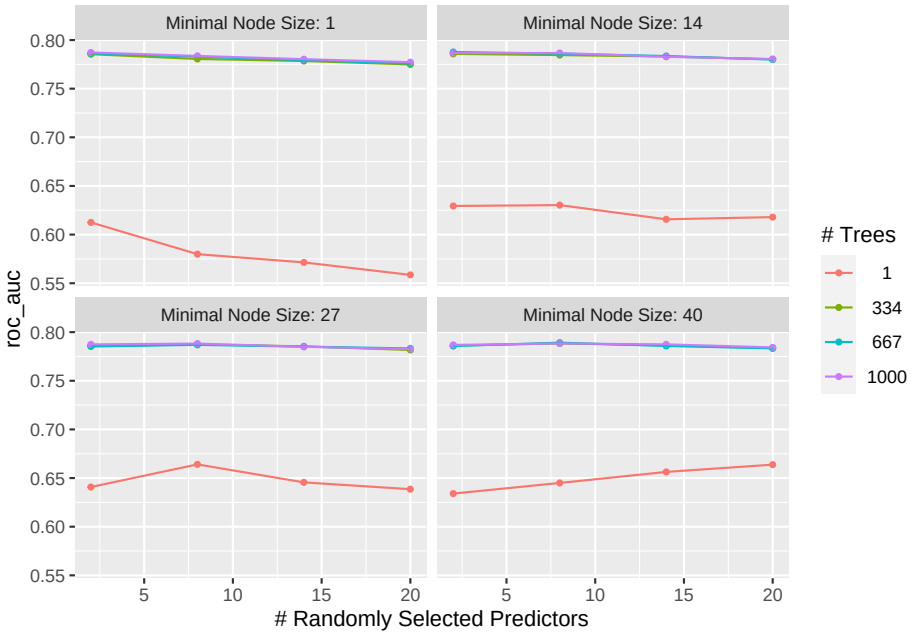
```
load(file = "model_fit/lda_tune.rda")
collect_metrics(tune_lda)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.852     5 0.00375 Preprocessor1_Model1
## 2 roc_auc  binary     0.797     5 0.00914 Preprocessor1_Model1
```

- Random forest

```
load(file = "model_fit/randomforest_tune.rda")

autoplot(rf_tune, metric = "roc_auc")
```

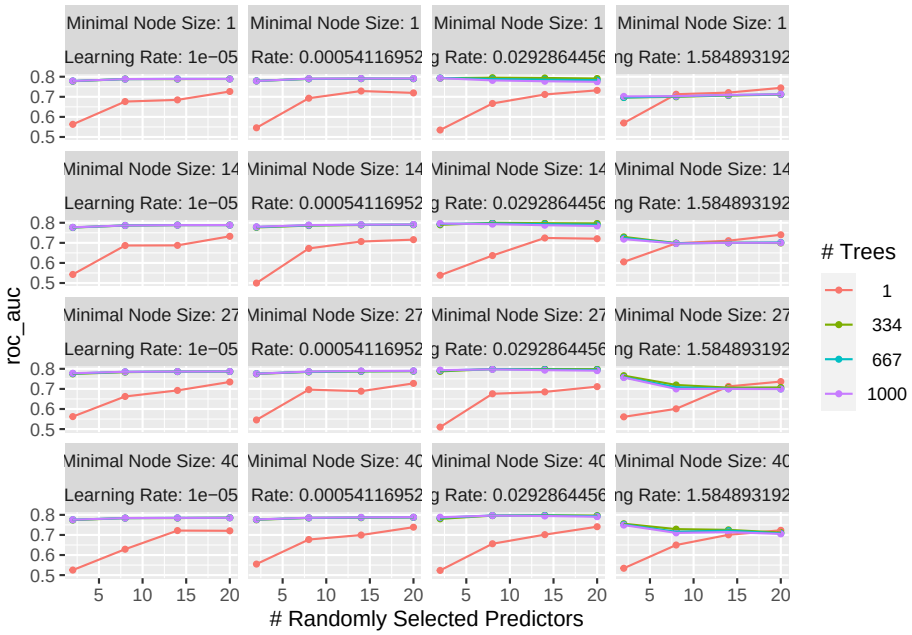



```
show_best(rf_tune, metric = "roc_auc") %>%
  dplyr::select(-.estimator, -.config)
```

```
## # A tibble: 5 x 7
##    mtry trees min_n .metric  mean     n std_err
##   <int> <int> <int> <chr>   <dbl> <int>   <dbl>
## 1     8   667    40 roc_auc 0.789     5 0.00977
## 2     8   334    40 roc_auc 0.788     5 0.0101
## 3     8  1000    40 roc_auc 0.788     5 0.00974
## 4     8  1000    27 roc_auc 0.788     5 0.00996
## 5     2   667    14 roc_auc 0.788     5 0.00969
```

- Boosted tree

```
load(file = "model_fit/bt_tune.rda")
autoplot(bt_tune, metric = "roc_auc")
```




```
show_best(bt_tune, metric = "roc_auc") %>% dplyr::select(-.estimator, -.config)
```

```
## # A tibble: 5 x 8
##    mtry trees min_n learn_rate .metric  mean     n std_err
##   <int> <int> <int>      <dbl> <chr>   <dbl> <int>   <dbl>
## 1     8   334    14     0.0293 roc_auc 0.800     5 0.00974
## 2    14   334    27     0.0293 roc_auc 0.799     5 0.00973
## 3     8   334    27     0.0293 roc_auc 0.799     5 0.00910
## 4    20   334    27     0.0293 roc_auc 0.798     5 0.00991
## 5     8   667    27     0.0293 roc_auc 0.798     5 0.00981
```

Combining all the performance results, I conclude that Boosted tree model is the best model, and random forest is almost as precise as logistic regression. In the next section, I apply these two models to test data and measure the prediction performance (by AUC).

Applying the Best Model(s) to the Test Set

Conclusion