CDC Diabetes

Bhavya Shri Meda, Mohamed Ndiaye & Bill Muchero

2024-06-15

Background

To help us compare our findings from the association rule model, we will use logistic regression to analyse our data to assess the relationship between lifestyle and diabetes in the US.

Libraries

```
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.1.1 --
## v broom        1.0.5     v rsample      1.2.0
## v dials        1.2.0     v tune         1.1.2
## v infer        1.0.5     v workflows    1.1.3
## v modeldata    1.3.0     v workflowsets 1.0.1
## v parsnip      1.1.1     v yardstick    1.3.0
## v recipes      1.0.9
## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Use suppressPackageStartupMessages() to eliminate package startup messages
```

```
library(readr)
library(stringr)
library(googledrive)
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
```

Importing Data

```
cdc <- read_csv("2015.csv")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## Rows: 441456 Columns: 330
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (7): IDATE, IMONTH, IDAY, IYEAR, PCDMDECN, EXACTOT1, EXACTOT2
## dbl (315): _STATE, FMONTH, DISPCODE, SEQNO, _PSU, CTELENUM, PVTRESD1, STATER...
## lgl   (8): COLGHOUS, LADULT, PAINACT2, QLMENTL2, QLSTRES2, QLHLTH2, ASERVIST...
##
## 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.
```

Since this is a large dataset, we are going to select the features that are of interest to the group. Some of the features that we will focus on include income, mental health, cholesterol and many others. Once we used the `select()` function to choose our features, we will rename our features so that they are easier to understand.

```
# Selecting variables of interest
cdc2015 <- cdc |>
  select('DIABETE3', '_RFHYPE5', 'TOLDHI2', '_CHOLCHK', '_BMI5','SMOKE100',
         'CVDSTRK3', '_MICHD', '_TOTINDA', '_FRTLT1', '_VEGLT1', '_RFDRHV5',
         'HLTHPLN1', 'MEDCOST', 'GENHLTH', 'MENTHLTH', 'PHYSHLTH', 'DIFFWALK',
        'SEX', '_AGEG5YR', 'EDUCA', 'INCOME2')

# Renaming the variables
cdc2015 <- cdc2015 |>
  dplyr::rename(Diabetes_012 = DIABETE3, HighBP = `_RFHYPE5`, HighChol = TOLDHI2, CholCheck = '_CHOLCHK
        BMI = '_BMI5', Smoker = SMOKE100, Stroke = CVDSTRK3, HeartDiseaseorAttack = '_MICHD',
         PhysActivity = '_TOTINDA', Fruits = '_FRTLT1', Veggies = '_VEGLT1', HvyAlcoholConsump = '_RFDR
```

```
        AnyHealthcare = 'HLTHPLN1', NoDocbcCost = MEDCOST, GenHlth = GENHLTH, MentHlth = MENTHLTH,
         PhysHlth = PHYSHLTH, DiffWalk = DIFFWALK, Sex = SEX, Age = '_AGEG5YR', Education = EDUCA,
        Income = INCOME2)
```

Data Preprocessing

To make sure our dataset is clean, we are going to eliminate duplicates and missing values to make sure we get the best results when we build and test our model

Data Cleaning

```
# Removing missing values
cdc2015 <- na.omit(cdc2015)

# Removing duplicates
cdc2015 <- distinct(cdc2015)
```

Selection of Variables

Now that our dataset is 'clean', we are going to reduce the dataset by selecting the features we are interested in investigating. Although many features are obvious causes of diabetes, we believe the features we have selected below are not the most obvious causes of diabetes.

```
cdc_five <- cdc2015 |>
  select('Diabetes_012','HighChol','Smoker','MentHlth','Education','Income')
```

Removing outliers

Below are the different values that are assigned. We are more interested in the values that provide clear answers for the sake of the study. For example, Diabetes_012 has the following values:

1 Yes 2 Yes, but female told only during pregnancy 3 No 4 No, pre-diabetes or borderline diabetes 7 Don't know/Not Sure 9 Refused The values 7 and 9 are considered not relevant for the dataset as we can't assume their true value.

```
# Removing 7 & 9 from Diabetes_012
cdc_five <- cdc_five |>
  filter(Diabetes_012 != 7 & Diabetes_012 != 9)
```

We now have 4 values left for Diabetes_012. Based on the BRFSS 2015 dataset, we can bin the values into 3 categories

Category 0: No Diabetes (values 2 and 3)

Category 1: Prediabetes (value 4)

Category 2: Diabetes (value 1)

Reference

1 Yes 2 Yes, but female told only during pregnancy 3 No 4 No, pre-diabetes or borderline diabetes The value 2 is not considered as a chronic diabetes as it happened only during pregnancy.

```r
# Replacing values for Category 0, Category 1, Category 2
cdc_five <- cdc_five |>
              mutate(Diabetes_012 = case_when(
              Diabetes_012 == 2 ~ 0,
              Diabetes_012 == 3 ~ 0,
              Diabetes_012 == 4 ~ 1,
              Diabetes_012 == 1 ~ 2,
              TRUE ~ Diabetes_012))

table(cdc_five$Diabetes_012)
```

```
##
##      0      1      2
## 282734   6516  49483
```

Below are the different values for MentHlth. The values mean how many days during the past 30 days was your mental health not good? We are more interested into values that provide clear answers for the sake of the study.

1 - 30 Number of days 88 None 77 Don't know/Not sure 99 Refused The values 77 and 99 are considered not relevant for the dataset as we can't assume their true value.

and the value 88 will be changed to 0 meaning no days of bad mental health in the last 30 days.

```r
# Removing 77 & 99 from MentHlth
cdc_five <- cdc_five |>
  filter(MentHlth != 77 & MentHlth != 99)

# Replacing 88 with 0
cdc_five <- cdc_five |>
              mutate(MentHlth = case_when(
              MentHlth == 88 ~ 0,
              TRUE ~ MentHlth))
```

Now we are going to do the same cleaning process for the remaining features (Smoker, Education, Income & Cholesterol)

```r
# Removing 7 & 9 from Smoker
cdc_five <- cdc_five |>
  filter(Smoker != 7 & Smoker != 9)

# Replacing 2 with 0
cdc_five <- cdc_five |>
              mutate(Smoker = case_when(
              Smoker == 2 ~ 0,
              TRUE ~ Smoker))
```

Cleaning the Education feature

```r
# Removing 7 & 9 from Education
cdc_five <- cdc_five |>
  filter(Education != 9)
```

Cleaning the Income feature

```
# Removing 77 & 99 from Income
cdc_five <- cdc_five |>
  filter(Income != 77 & Income != 99)
```

Cleaning Cholesterol feature

```
# Removing 7 & 9 from Cholesterol
cdc_five <- cdc_five |>
  filter(HighChol != 7 & HighChol != 9)

# Replacing 2 with 0
cdc_five <- cdc_five |>
              mutate(HighChol = case_when(
              HighChol == 2 ~ 0,
              TRUE ~ HighChol))

# Changing the Diabetes values from numeric to string
cdc_five <- cdc_five  |>
              mutate(Diabetes_012 = case_when(
                      Diabetes_012  == 0 ~ "No Diabetes",
                      Diabetes_012 == 1 ~ "Pre-Diabetes",
                      Diabetes_012 == 2 ~ "Diabetes",
                         TRUE~ "other"))
```

Splitting the Data

To build our model, we will first split our dataset into a training set and a testing set. This will allow us to train and test our model to ensure accuracy.

```
#Changing the Diabetes_012 feature to a factor
cdc_five$Diabetes_012 <- as.factor(cdc_five$Diabetes_012)

# Split data into train and test
set.seed(12)
split <- initial_split(cdc_five, prop = 0.8, strata = Diabetes_012)
train <- split |>
          training()
test <- split |>
        testing()
```

Building a Logistic Regression Model

To compare our findings from the association rule model, we will use logistic regression to build a model that can predict if someone has diabetes or not when given certain features.

Since our `Diabetes_012` feature has more than two categorical variables, we will use a multinomial logistic regression for our model.

```
# Train a logistic regression model
model <- multinom_reg(penalty = 0.1, mixture = 1) |>
  set_engine("glmnet") |>
  set_mode("classification") |>
```

```
  fit(Diabetes_012 ~ ., data = train)

# Model summary
tidy(model)
```

```
## # A tibble: 18 x 4
##    class        term        estimate penalty
##    <chr>        <chr>          <dbl>   <dbl>
##  1 Diabetes     (Intercept)   0.0927     0.1
##  2 Diabetes     HighChol      0          0.1
##  3 Diabetes     Smoker        0          0.1
##  4 Diabetes     MentHlth      0          0.1
##  5 Diabetes     Education     0          0.1
##  6 Diabetes     Income        0          0.1
##  7 No Diabetes  (Intercept)   1.85       0.1
##  8 No Diabetes  HighChol      0          0.1
##  9 No Diabetes  Smoker        0          0.1
## 10 No Diabetes  MentHlth      0          0.1
## 11 No Diabetes  Education     0          0.1
## 12 No Diabetes  Income        0          0.1
## 13 Pre-Diabetes (Intercept)  -1.95       0.1
## 14 Pre-Diabetes HighChol      0          0.1
## 15 Pre-Diabetes Smoker        0          0.1
## 16 Pre-Diabetes MentHlth      0          0.1
## 17 Pre-Diabetes Education     0          0.1
## 18 Pre-Diabetes Income        0          0.1
```

Now that we have our model, we will now try to make predictions. To make these predictions, we will use two different classes. In this instance `type = "class"` will allow us to get a target value ("yes" or "no") for each observation. On the other hand, `type = "prob"` will give us the probability of each target value for each observation.

```
# Creating Class Predictions
pred_class <- predict(model, new_data = test, type = "class")

# Creating Class Probabilities
pred_proba <- predict(model, new_data = test, type = "prob")
```

To evaluate our model, we will use the `accuracy()` function.

```
results <- test |>
            select(Diabetes_012) |>
            bind_cols(pred_class, pred_proba)

# Evaluating the model's accuracy
accuracy(results, truth = Diabetes_012, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.838
```

Without much domain knowledge, its hard to say if our model's estimate score is a good or reflective of how accurate our model is. However, if we are going by assumptions, one can say the estimate result means our model is pretty accurate when determining if someone either has diabetes or not, or if they are pre-diabetic.

Hyperparameter Tuning

To make our model better, we are going to use the hyperparameter tuning technique. This will allow us to run the model several times using different values. The results will allow us to know which mixture and penalty arguments will give us the best predictions.

```
# Define the logistic regression model with penalty and mixture hyperparameters
multinom_reg_model <- multinom_reg(mixture = tune(), penalty = tune(), engine = "glmnet")

# Define the grid search for the hyperparameters
grid <- grid_regular(mixture(), penalty(), levels = c(mixture = 4, penalty = 3))

# Define the workflow for the model
multinom_wf <- workflow() |>
  add_model(multinom_reg_model) |>
  add_formula(Diabetes_012 ~ .)

# Define the resampling method for the grid search
folds <- vfold_cv(train, v = 5)

# Tune the hyperparameters using the grid search
mulitnom_tuned <- tune_grid(
  multinom_wf,
  resamples = folds,
  grid = grid,
  control = control_grid(save_pred = TRUE)
)

select_best(mulitnom_tuned, metric = "roc_auc")
```

```
## # A tibble: 1 x 3
##        penalty mixture .config
##          <dbl>   <dbl> <chr>
## 1 0.0000000001       0 Preprocessor1_Model01
```

Now that we have our best penalty and mixture values, we now know which values will make our model work best. In this case, the penalty value needs to be 1e-10 and the mixture value needs to be 0. Below we shall build the final model using the penalty and mixture values we got above.

Once we have our model, we will create a confusion matrix to evaluate our model.

```
# Fit the model using the optimal hyperparameters
multinom_final <- multinom_reg(penalty = 1e-10, mixture = 0) |>
                  set_engine("glmnet") |>
                  set_mode("classification") |>
                  fit(Diabetes_012~., data = train)

# Evaluate the model performance on the testing set
pred_class <- predict(multinom_final,
```

```
                      new_data = test,
                      type = "class")
results <- test |>
  select(Diabetes_012) |>
  bind_cols(pred_class, pred_proba)

# Create confusion matrix
conf_mat(results, truth = Diabetes_012,
         estimate = .pred_class)
```

```
##               Truth
## Prediction     Diabetes No Diabetes Pre-Diabetes
##   Diabetes           19          14            2
##   No Diabetes      8128       47291         1021
##   Pre-Diabetes        0           0            0
```

From the results above, our model did a better job predicting if someone had diabetes given the various features. However, it did poorly in determining if some did not have diabetes. This means we will need to look into how we can make this model better given the model's errors and weaknesses.

```
precision(results, truth = Diabetes_012,
          estimate = .pred_class)
```

```
## Warning: While computing multiclass `precision()`, some levels had no predicted events
## (i.e. `true_positive + false_positive = 0`).
## Precision is undefined in this case, and those levels will be removed from the
## averaged result.
## Note that the following number of true events actually occurred for each
## problematic event level:
## 'Pre-Diabetes': 1023
```

```
## # A tibble: 1 x 3
##   .metric   .estimator .estimate
##   <chr>     <chr>          <dbl>
## 1 precision macro          0.690
```

Now to assess our model's precision, we will use the the `precision()` function again. In this case, our final model has a lower estimate value than our original model. This might be the a better reflection of how precise our model is given the errors/false predictions we got when we did the confusion matrix.

In summation, our model has errors/weaknesses that need to be fixed. Given our lack of domain knowledge, its hard to say how precise our model is. In addition, its hard to say our model's estimate value represents how good our model is because the value depends on precision and the domain as well.