P8451\_HW6\_ML

Ruixi Li

2024-02-20

library(NHANES)  
library(tidyverse)  
library(caret)  
library(rpart.plot)  
library(pROC)

# 1. Load data

data(NHANES)

# 2. select features and partition data

* Although classification tree and support vector machine are both non-parametric method and don’t assume the independence of features, interpretation might be faulty since dependence/correlation between features could be intrepreted as an interaction effect. This will not effect prediction. So, I would drop weight and height, two of which were used to calculate BMI. But theoretically, BMI is more correlated with diabetes and pre-diabetes. Although it’s believed that BMI and physical activity are correlated, I think physical activity have some unexplained effect towards diabetes and should be kept as a predictor.

# feature selection and remove duplicate rows  
NHANES\_tailored = NHANES|>  
 select(Age, Race1, Education, HHIncome, Pulse, Diabetes, BMI, PhysActive, Smoke100) |> janitor::clean\_names() |> distinct()  
  
# investigation and light cleaning  
NHANES\_tailored|> skimr::skim()

Data summary

|  |  |
| --- | --- |
| Name | NHANES\_tailored |
| Number of rows | 6627 |
| Number of columns | 9 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 6 |
| numeric | 3 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

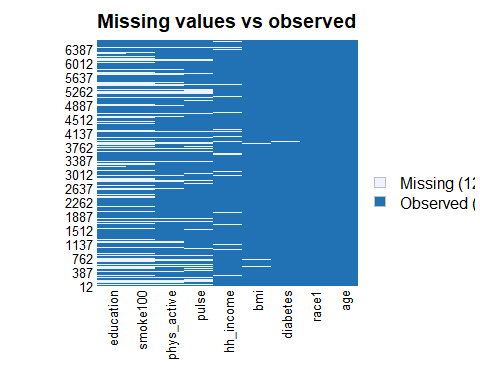
**Variable type: factor**

| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| --- | --- | --- | --- | --- | --- |
| race1 | 0 | 1.00 | FALSE | 5 | Whi: 3678, Bla: 1013, Mex: 837, Oth: 610 |
| education | 1982 | 0.70 | FALSE | 5 | Som: 1423, Col: 1246, Hig: 993, 9 -: 624 |
| hh\_income | 574 | 0.91 | FALSE | 12 | mor: 1275, 250: 674, 750: 666, 350: 601 |
| diabetes | 50 | 0.99 | FALSE | 2 | No: 6025, Yes: 552 |
| phys\_active | 1182 | 0.82 | FALSE | 2 | Yes: 2972, No: 2473 |
| smoke100 | 1973 | 0.70 | FALSE | 2 | No: 2598, Yes: 2056 |

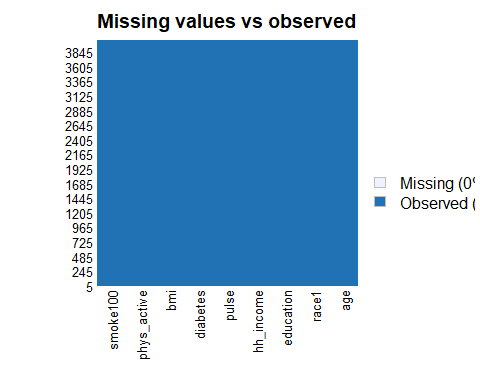
**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| age | 0 | 1.00 | 36.25 | 22.76 | 0.00 | 16.00 | 35.00 | 54.0 | 80.00 | ▇▇▆▆▅ |
| pulse | 975 | 0.85 | 73.66 | 12.25 | 40.00 | 64.00 | 72.00 | 82.0 | 136.00 | ▂▇▃▁▁ |
| bmi | 156 | 0.98 | 26.49 | 7.54 | 12.88 | 21.24 | 25.82 | 30.6 | 81.25 | ▇▆▁▁▁ |

NHANES\_tailored |> Amelia::missmap(main = "Missing values vs observed")



NHANES\_tailored = NHANES\_tailored |> drop\_na()  
  
NHANES\_tailored |> Amelia::missmap(main = "Missing values vs observed")



#tidyverse way to create data partition   
set.seed(123)  
  
train.indices = NHANES\_tailored |>  
 pull(diabetes) |>  
 createDataPartition(p = 0.7, list = FALSE)  
  
train.data = NHANES\_tailored |>  
 slice(train.indices)  
  
test.data = NHANES\_tailored |>  
 slice(-train.indices)  
  
summary(NHANES\_tailored$diabetes)

## No Yes   
## 3562 478

# Note that data are slightly unbalanced.

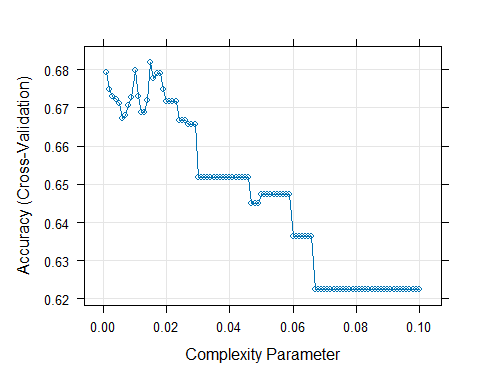
# 3. Model building

## classification tree

set.seed(123)  
#Creating 10-fold cross-validation and using down-sampling because of imbalance in data  
control= trainControl(method="cv", number=10, sampling="down", classProbs = T)  
  
# Up-sampling is preferred when the dataset is small, and data loss is a concern, while down-sampling might be more appropriate for large datasets where training time and resource utilization are significant considerations.  
  
  
# classification tree  
  
modelLookup("rpart")

## model parameter label forReg forClass probModel  
## 1 rpart cp Complexity Parameter TRUE TRUE TRUE

#Specify tuneGrid so caret explores wider variety of cp-values  
set.seed(123)  
  
#Create different values of cp to try  
cp.grid=expand.grid(cp=seq(0.001,0.1, by=0.001))  
tree=train(  
 diabetes ~ .,   
 data=train.data,   
 method="rpart",   
 trControl=control,   
 tuneGrid=cp.grid,  
 preProcess=c("center", "scale")  
 )  
  
plot(tree, uniform=TRUE)



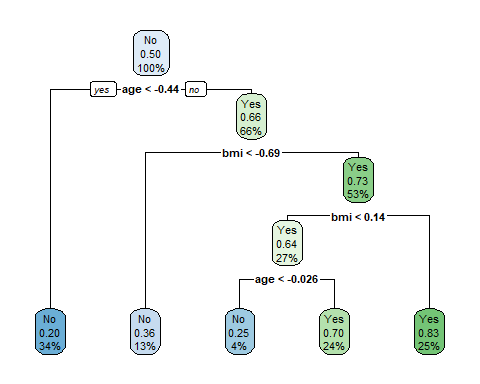
tree$bestTune

## cp  
## 15 0.015

tree$results

## cp Accuracy Kappa AccuracySD KappaSD  
## 1 0.001 0.6793960 0.1860360 0.02719124 0.04118558  
## 2 0.002 0.6747635 0.1853100 0.04281398 0.04572766  
## 3 0.003 0.6729967 0.1819731 0.03982714 0.04278820  
## 4 0.004 0.6723000 0.1887908 0.03563237 0.03598823  
## 5 0.005 0.6712424 0.1955107 0.03547253 0.03282907  
## 6 0.006 0.6673567 0.2031245 0.03549025 0.03725830  
## 7 0.007 0.6680609 0.2013645 0.03682914 0.04713440  
## 8 0.008 0.6705432 0.2021728 0.03090022 0.04516174  
## 9 0.009 0.6726734 0.2095178 0.04396298 0.04778860  
## 10 0.010 0.6797880 0.2211364 0.05292402 0.04454600  
## 11 0.011 0.6730742 0.2179533 0.05288914 0.04622700  
## 12 0.012 0.6688439 0.2148097 0.05341186 0.04427196  
## 13 0.013 0.6688439 0.2148097 0.05341186 0.04427196  
## 14 0.014 0.6720129 0.2126418 0.05122334 0.04005831  
## 15 0.015 0.6818932 0.2198545 0.04917967 0.03913933  
## 16 0.016 0.6776379 0.2174527 0.04662236 0.03817543  
## 17 0.017 0.6790450 0.2216821 0.05166848 0.04354135  
## 18 0.018 0.6790450 0.2216821 0.05166848 0.04354135  
## 19 0.019 0.6748334 0.2210778 0.06459447 0.04535252  
## 20 0.020 0.6716433 0.2126388 0.05017054 0.03971128  
## 21 0.021 0.6716433 0.2126388 0.05017054 0.03971128  
## 22 0.022 0.6716433 0.2126388 0.05017054 0.03971128  
## 23 0.023 0.6716433 0.2126388 0.05017054 0.03971128  
## 24 0.024 0.6666963 0.2063537 0.05135261 0.03120278  
## 25 0.025 0.6666963 0.2063537 0.05135261 0.03120278  
## 26 0.026 0.6666963 0.2063537 0.05135261 0.03120278  
## 27 0.027 0.6656161 0.2023288 0.05837131 0.03261610  
## 28 0.028 0.6656161 0.2023288 0.05837131 0.03261610  
## 29 0.029 0.6656161 0.2023288 0.05837131 0.03261610  
## 30 0.030 0.6518352 0.1989322 0.05544228 0.03387741  
## 31 0.031 0.6518352 0.1989322 0.05544228 0.03387741  
## 32 0.032 0.6518352 0.1989322 0.05544228 0.03387741  
## 33 0.033 0.6518352 0.1989322 0.05544228 0.03387741  
## 34 0.034 0.6518352 0.1989322 0.05544228 0.03387741  
## 35 0.035 0.6518352 0.1989322 0.05544228 0.03387741  
## 36 0.036 0.6518352 0.1989322 0.05544228 0.03387741  
## 37 0.037 0.6518352 0.1989322 0.05544228 0.03387741  
## 38 0.038 0.6518352 0.1989322 0.05544228 0.03387741  
## 39 0.039 0.6518352 0.1989322 0.05544228 0.03387741  
## 40 0.040 0.6518352 0.1989322 0.05544228 0.03387741  
## 41 0.041 0.6518352 0.1989322 0.05544228 0.03387741  
## 42 0.042 0.6518352 0.1989322 0.05544228 0.03387741  
## 43 0.043 0.6518352 0.1989322 0.05544228 0.03387741  
## 44 0.044 0.6518352 0.1989322 0.05544228 0.03387741  
## 45 0.045 0.6518352 0.1989322 0.05544228 0.03387741  
## 46 0.046 0.6518352 0.1989322 0.05544228 0.03387741  
## 47 0.047 0.6450976 0.1935822 0.04565720 0.02412143  
## 48 0.048 0.6450976 0.1935822 0.04565720 0.02412143  
## 49 0.049 0.6450976 0.1935822 0.04565720 0.02412143  
## 50 0.050 0.6472178 0.1911850 0.04513227 0.02694270  
## 51 0.051 0.6472178 0.1911850 0.04513227 0.02694270  
## 52 0.052 0.6472178 0.1911850 0.04513227 0.02694270  
## 53 0.053 0.6472178 0.1911850 0.04513227 0.02694270  
## 54 0.054 0.6472178 0.1911850 0.04513227 0.02694270  
## 55 0.055 0.6472178 0.1911850 0.04513227 0.02694270  
## 56 0.056 0.6472178 0.1911850 0.04513227 0.02694270  
## 57 0.057 0.6472178 0.1911850 0.04513227 0.02694270  
## 58 0.058 0.6472178 0.1911850 0.04513227 0.02694270  
## 59 0.059 0.6472178 0.1911850 0.04513227 0.02694270  
## 60 0.060 0.6363023 0.1843559 0.04092393 0.02672190  
## 61 0.061 0.6363023 0.1843559 0.04092393 0.02672190  
## 62 0.062 0.6363023 0.1843559 0.04092393 0.02672190  
## 63 0.063 0.6363023 0.1843559 0.04092393 0.02672190  
## 64 0.064 0.6363023 0.1843559 0.04092393 0.02672190  
## 65 0.065 0.6363023 0.1843559 0.04092393 0.02672190  
## 66 0.066 0.6363023 0.1843559 0.04092393 0.02672190  
## 67 0.067 0.6224725 0.1803389 0.04592717 0.02904421  
## 68 0.068 0.6224725 0.1803389 0.04592717 0.02904421  
## 69 0.069 0.6224725 0.1803389 0.04592717 0.02904421  
## 70 0.070 0.6224725 0.1803389 0.04592717 0.02904421  
## 71 0.071 0.6224725 0.1803389 0.04592717 0.02904421  
## 72 0.072 0.6224725 0.1803389 0.04592717 0.02904421  
## 73 0.073 0.6224725 0.1803389 0.04592717 0.02904421  
## 74 0.074 0.6224725 0.1803389 0.04592717 0.02904421  
## 75 0.075 0.6224725 0.1803389 0.04592717 0.02904421  
## 76 0.076 0.6224725 0.1803389 0.04592717 0.02904421  
## 77 0.077 0.6224725 0.1803389 0.04592717 0.02904421  
## 78 0.078 0.6224725 0.1803389 0.04592717 0.02904421  
## 79 0.079 0.6224725 0.1803389 0.04592717 0.02904421  
## 80 0.080 0.6224725 0.1803389 0.04592717 0.02904421  
## 81 0.081 0.6224725 0.1803389 0.04592717 0.02904421  
## 82 0.082 0.6224725 0.1803389 0.04592717 0.02904421  
## 83 0.083 0.6224725 0.1803389 0.04592717 0.02904421  
## 84 0.084 0.6224725 0.1803389 0.04592717 0.02904421  
## 85 0.085 0.6224725 0.1803389 0.04592717 0.02904421  
## 86 0.086 0.6224725 0.1803389 0.04592717 0.02904421  
## 87 0.087 0.6224725 0.1803389 0.04592717 0.02904421  
## 88 0.088 0.6224725 0.1803389 0.04592717 0.02904421  
## 89 0.089 0.6224725 0.1803389 0.04592717 0.02904421  
## 90 0.090 0.6224725 0.1803389 0.04592717 0.02904421  
## 91 0.091 0.6224725 0.1803389 0.04592717 0.02904421  
## 92 0.092 0.6224725 0.1803389 0.04592717 0.02904421  
## 93 0.093 0.6224725 0.1803389 0.04592717 0.02904421  
## 94 0.094 0.6224725 0.1803389 0.04592717 0.02904421  
## 95 0.095 0.6224725 0.1803389 0.04592717 0.02904421  
## 96 0.096 0.6224725 0.1803389 0.04592717 0.02904421  
## 97 0.097 0.6224725 0.1803389 0.04592717 0.02904421  
## 98 0.098 0.6224725 0.1803389 0.04592717 0.02904421  
## 99 0.099 0.6224725 0.1803389 0.04592717 0.02904421  
## 100 0.100 0.6224725 0.1803389 0.04592717 0.02904421

#Plot new "best" tree  
tree$finalModel |>  
 rpart.plot()



#Example variable importance in model  
varImp(tree)

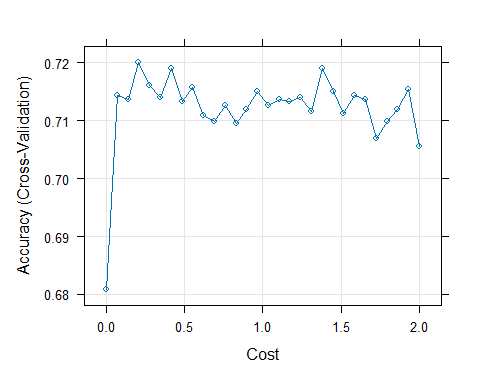
## rpart variable importance  
##   
## only 20 most important variables shown (out of 26)  
##   
## Overall  
## age 100.000  
## bmi 65.948  
## phys\_activeYes 34.379  
## educationCollege Grad 26.661  
## hh\_incomemore 99999 20.201  
## race1White 7.320  
## pulse 2.347  
## `educationHigh School` 0.000  
## smoke100Yes 0.000  
## race1Hispanic 0.000  
## race1Other 0.000  
## `hh\_income20000-24999` 0.000  
## `hh\_income55000-64999` 0.000  
## `hh\_incomemore 99999` 0.000  
## `educationCollege Grad` 0.000  
## `hh\_income 5000-9999` 0.000  
## `hh\_income65000-74999` 0.000  
## `educationSome College` 0.000  
## `hh\_income10000-14999` 0.000  
## `hh\_income25000-34999` 0.000

## Support Vector Machine

library(e1071)  
modelLookup("svmLinear")

## model parameter label forReg forClass probModel  
## 1 svmLinear C Cost TRUE TRUE TRUE

set.seed(123)  
#Incorporate different values for cost (C)  
svm=train(  
 diabetes ~ .,   
 data=train.data,   
 method="svmLinear",   
 trControl=control,   
 preProcess=c("center", "scale"),   
 tuneGrid=expand.grid(C=seq(0.001,2, length=30))  
 )  
  
#Visualize accuracy versus values of C  
plot(svm)



svm$bestTune

## C  
## 4 0.2077931

#See information about final model  
svm$finalModel

## Support Vector Machine object of class "ksvm"   
##   
## SV type: C-svc (classification)   
## parameter : cost C = 0.207793103448276   
##   
## Linear (vanilla) kernel function.   
##   
## Number of Support Vectors : 408   
##   
## Objective Function Value : -81.1628   
## Training error : 0.232836   
## Probability model included.

## Logistic Regression

set.seed(123)  
  
logistic = train(diabetes ~ .,   
 data = train.data, method = "glm",  
 family = "binomial",  
 trControl = control,  
 preProcess=c("center", "scale"))

# 4. Model selection and evaluation

#Obtain metrics of accuracy from training  
confusionMatrix(tree)

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 59.3 3.0  
## Yes 28.8 8.9  
##   
## Accuracy (average) : 0.6819

confusionMatrix(svm)

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 62.8 2.6  
## Yes 25.4 9.2  
##   
## Accuracy (average) : 0.72

confusionMatrix(logistic)

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 62.9 2.8  
## Yes 25.2 9.0  
##   
## Accuracy (average) : 0.7193

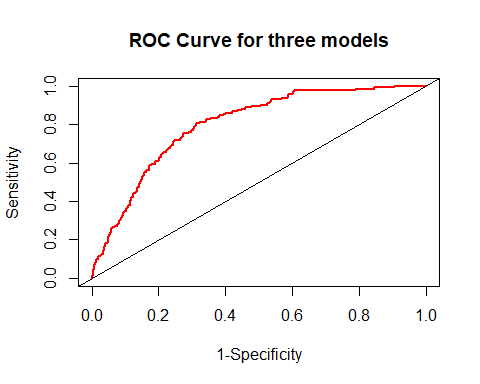
* Since the accuracy for Support Vector Machine is the highest(0.7213), I would chose Support Vector Machine as my optimal model.

## Support Vector Machine

#Create predictions in test set  
pred.svm = svm |>  
 predict(test.data)  
  
eval.results=confusionMatrix(pred.svm, test.data$diabetes, positive = "Yes")  
print(eval.results)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 733 29  
## Yes 335 114  
##   
## Accuracy : 0.6994   
## 95% CI : (0.6727, 0.7251)  
## No Information Rate : 0.8819   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.251   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.79720   
## Specificity : 0.68633   
## Pos Pred Value : 0.25390   
## Neg Pred Value : 0.96194   
## Prevalence : 0.11808   
## Detection Rate : 0.09414   
## Detection Prevalence : 0.37077   
## Balanced Accuracy : 0.74177   
##   
## 'Positive' Class : Yes   
##

#Create predictions as probabilities on test set   
pred.svm.prob = svm |>   
 predict(test.data, type = "prob")  
  
#Another potential evaluation: Area under the Receiver Operating Curve (AUROC)  
analysis = roc(response=test.data$diabetes, predictor=pred.svm.prob[,2])  
plot(1-analysis$specificities,analysis$sensitivities,type="l",  
ylab="Sensitivity",xlab="1-Specificity",col="red",lwd=2,  
main = "ROC Curve for three models")  
abline(a=0,b=1)



* The model appears to have a fair capacity to identify the positive class(sensitivity/recall=79.72%) but struggles with precision(precision=25.39%), indicating a high rate of false positives. This is also suggested by the low positive predictive value. The model is not outperforming a naive classifier that would predict the most frequent class for all cases, as evidenced by the comparison to the NIR and the associated p-value. The balanced accuracy is 74.18%, which is a better metric than accuracy for imbalanced datasets.

# 5. Ethical consideration of using race as a predictor

* Including race as a predictor in disease prediction models raises significant ethical considerations, reflecting broader societal issues around equity, discrimination, and the interpretation of genetic versus socio-environmental determinants of health. It may divert attention from underlying causes of health disparities, such as socio-economic status, physical activity and BMI which are more directly actionable.