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

Data summary

|  |  |
| --- | --- |
| Name | NHANES\_tailored |
| Number of rows | 10000 |
| Number of columns | 9 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 6 |
| numeric | 3 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

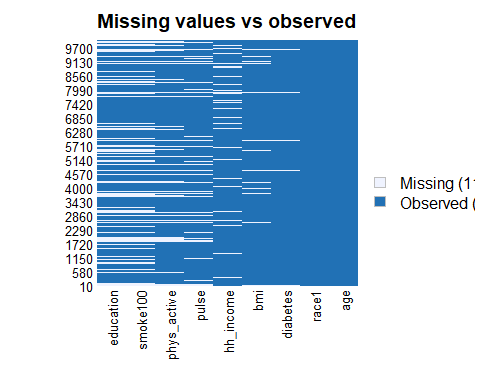
**Variable type: factor**

| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| --- | --- | --- | --- | --- | --- |
| race1 | 0 | 1.00 | FALSE | 5 | Whi: 6372, Bla: 1197, Mex: 1015, Oth: 806 |
| education | 2779 | 0.72 | FALSE | 5 | Som: 2267, Col: 2098, Hig: 1517, 9 -: 888 |
| hh\_income | 811 | 0.92 | FALSE | 12 | mor: 2220, 750: 1084, 250: 958, 350: 863 |
| diabetes | 142 | 0.99 | FALSE | 2 | No: 9098, Yes: 760 |
| phys\_active | 1674 | 0.83 | FALSE | 2 | Yes: 4649, No: 3677 |
| smoke100 | 2765 | 0.72 | FALSE | 2 | No: 4024, Yes: 3211 |

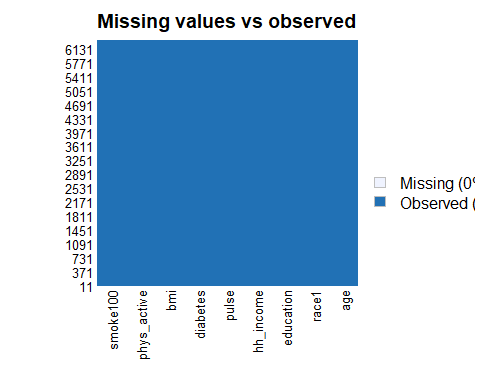
**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| age | 0 | 1.00 | 36.74 | 22.40 | 0.00 | 17.00 | 36.00 | 54.00 | 80.00 | ▇▇▇▆▅ |
| pulse | 1437 | 0.86 | 73.56 | 12.16 | 40.00 | 64.00 | 72.00 | 82.00 | 136.00 | ▂▇▃▁▁ |
| bmi | 366 | 0.96 | 26.66 | 7.38 | 12.88 | 21.58 | 25.98 | 30.89 | 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   
## 5697 659

# Note that data are slightly unbalanced.

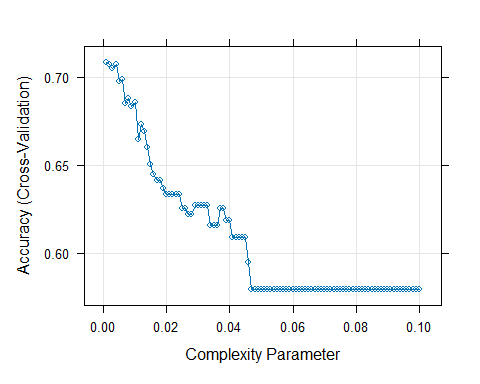
# 3. Model building

## 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)



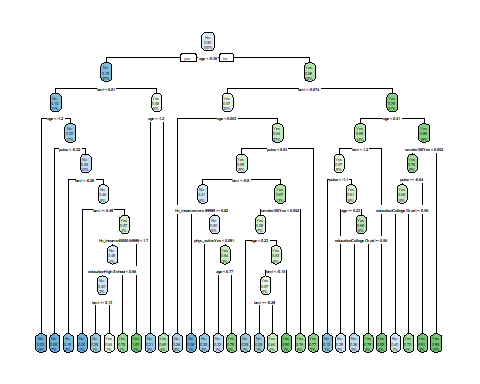
tree$bestTune

## cp  
## 1 0.001

tree$results

## cp Accuracy Kappa AccuracySD KappaSD  
## 1 0.001 0.7082998 0.2312504 0.02743510 0.02708708  
## 2 0.002 0.7071889 0.2339098 0.02987065 0.02496368  
## 3 0.003 0.7049255 0.2341159 0.02555511 0.03274315  
## 4 0.004 0.7071682 0.2340354 0.03759775 0.03602215  
## 5 0.005 0.6972755 0.2279772 0.03799301 0.04103330  
## 6 0.006 0.6984016 0.2283836 0.03826350 0.04096335  
## 7 0.007 0.6849295 0.2191887 0.04105887 0.04332392  
## 8 0.008 0.6880776 0.2216048 0.04167225 0.04245007  
## 9 0.009 0.6835670 0.2246480 0.06116510 0.05599001  
## 10 0.010 0.6855895 0.2201708 0.06305111 0.05121235  
## 11 0.011 0.6647037 0.2066997 0.06891341 0.05395558  
## 12 0.012 0.6729997 0.2119632 0.06527618 0.04919007  
## 13 0.013 0.6691744 0.2094563 0.07075901 0.05090370  
## 14 0.014 0.6601856 0.2019834 0.06402596 0.04152132  
## 15 0.015 0.6505227 0.2000445 0.04612262 0.03784308  
## 16 0.016 0.6444507 0.1922842 0.05135597 0.03973121  
## 17 0.017 0.6410880 0.1884117 0.05410448 0.04031430  
## 18 0.018 0.6410880 0.1884117 0.05410448 0.04031430  
## 19 0.019 0.6366037 0.1862113 0.05450056 0.04117729  
## 20 0.020 0.6332430 0.1793695 0.04712269 0.04534437  
## 21 0.021 0.6332430 0.1793695 0.04712269 0.04534437  
## 22 0.022 0.6332430 0.1793695 0.04712269 0.04534437  
## 23 0.023 0.6332430 0.1793695 0.04712269 0.04534437  
## 24 0.024 0.6332430 0.1793695 0.04712269 0.04534437  
## 25 0.025 0.6251349 0.1754049 0.05806398 0.04818121  
## 26 0.026 0.6251349 0.1754049 0.05806398 0.04818121  
## 27 0.027 0.6217641 0.1735868 0.05948664 0.04798737  
## 28 0.028 0.6217641 0.1735868 0.05948664 0.04798737  
## 29 0.029 0.6269443 0.1755479 0.05187012 0.04626204  
## 30 0.030 0.6269443 0.1755479 0.05187012 0.04626204  
## 31 0.031 0.6269443 0.1755479 0.05187012 0.04626204  
## 32 0.032 0.6269443 0.1755479 0.05187012 0.04626204  
## 33 0.033 0.6269443 0.1755479 0.05187012 0.04626204  
## 34 0.034 0.6157336 0.1714351 0.05029393 0.04630462  
## 35 0.035 0.6157336 0.1714351 0.05029393 0.04630462  
## 36 0.036 0.6157336 0.1714351 0.05029393 0.04630462  
## 37 0.037 0.6256212 0.1753831 0.04430491 0.04631499  
## 38 0.038 0.6256212 0.1753831 0.04430491 0.04631499  
## 39 0.039 0.6184463 0.1707069 0.05189584 0.05003576  
## 40 0.040 0.6184463 0.1707069 0.05189584 0.05003576  
## 41 0.041 0.6090081 0.1640054 0.04768762 0.04897937  
## 42 0.042 0.6090081 0.1640054 0.04768762 0.04897937  
## 43 0.043 0.6090081 0.1640054 0.04768762 0.04897937  
## 44 0.044 0.6090081 0.1640054 0.04768762 0.04897937  
## 45 0.045 0.6090081 0.1640054 0.04768762 0.04897937  
## 46 0.046 0.5948508 0.1546075 0.05011543 0.04583445  
## 47 0.047 0.5793452 0.1431098 0.03648829 0.02745343  
## 48 0.048 0.5793452 0.1431098 0.03648829 0.02745343  
## 49 0.049 0.5793452 0.1431098 0.03648829 0.02745343  
## 50 0.050 0.5793452 0.1431098 0.03648829 0.02745343  
## 51 0.051 0.5793452 0.1431098 0.03648829 0.02745343  
## 52 0.052 0.5793452 0.1431098 0.03648829 0.02745343  
## 53 0.053 0.5793452 0.1431098 0.03648829 0.02745343  
## 54 0.054 0.5793452 0.1431098 0.03648829 0.02745343  
## 55 0.055 0.5793452 0.1431098 0.03648829 0.02745343  
## 56 0.056 0.5793452 0.1431098 0.03648829 0.02745343  
## 57 0.057 0.5793452 0.1431098 0.03648829 0.02745343  
## 58 0.058 0.5793452 0.1431098 0.03648829 0.02745343  
## 59 0.059 0.5793452 0.1431098 0.03648829 0.02745343  
## 60 0.060 0.5793452 0.1431098 0.03648829 0.02745343  
## 61 0.061 0.5793452 0.1431098 0.03648829 0.02745343  
## 62 0.062 0.5793452 0.1431098 0.03648829 0.02745343  
## 63 0.063 0.5793452 0.1431098 0.03648829 0.02745343  
## 64 0.064 0.5793452 0.1431098 0.03648829 0.02745343  
## 65 0.065 0.5793452 0.1431098 0.03648829 0.02745343  
## 66 0.066 0.5793452 0.1431098 0.03648829 0.02745343  
## 67 0.067 0.5793452 0.1431098 0.03648829 0.02745343  
## 68 0.068 0.5793452 0.1431098 0.03648829 0.02745343  
## 69 0.069 0.5793452 0.1431098 0.03648829 0.02745343  
## 70 0.070 0.5793452 0.1431098 0.03648829 0.02745343  
## 71 0.071 0.5793452 0.1431098 0.03648829 0.02745343  
## 72 0.072 0.5793452 0.1431098 0.03648829 0.02745343  
## 73 0.073 0.5793452 0.1431098 0.03648829 0.02745343  
## 74 0.074 0.5793452 0.1431098 0.03648829 0.02745343  
## 75 0.075 0.5793452 0.1431098 0.03648829 0.02745343  
## 76 0.076 0.5793452 0.1431098 0.03648829 0.02745343  
## 77 0.077 0.5793452 0.1431098 0.03648829 0.02745343  
## 78 0.078 0.5793452 0.1431098 0.03648829 0.02745343  
## 79 0.079 0.5793452 0.1431098 0.03648829 0.02745343  
## 80 0.080 0.5793452 0.1431098 0.03648829 0.02745343  
## 81 0.081 0.5793452 0.1431098 0.03648829 0.02745343  
## 82 0.082 0.5793452 0.1431098 0.03648829 0.02745343  
## 83 0.083 0.5793452 0.1431098 0.03648829 0.02745343  
## 84 0.084 0.5793452 0.1431098 0.03648829 0.02745343  
## 85 0.085 0.5793452 0.1431098 0.03648829 0.02745343  
## 86 0.086 0.5793452 0.1431098 0.03648829 0.02745343  
## 87 0.087 0.5793452 0.1431098 0.03648829 0.02745343  
## 88 0.088 0.5793452 0.1431098 0.03648829 0.02745343  
## 89 0.089 0.5793452 0.1431098 0.03648829 0.02745343  
## 90 0.090 0.5793452 0.1431098 0.03648829 0.02745343  
## 91 0.091 0.5793452 0.1431098 0.03648829 0.02745343  
## 92 0.092 0.5793452 0.1431098 0.03648829 0.02745343  
## 93 0.093 0.5793452 0.1431098 0.03648829 0.02745343  
## 94 0.094 0.5793452 0.1431098 0.03648829 0.02745343  
## 95 0.095 0.5793452 0.1431098 0.03648829 0.02745343  
## 96 0.096 0.5793452 0.1431098 0.03648829 0.02745343  
## 97 0.097 0.5793452 0.1431098 0.03648829 0.02745343  
## 98 0.098 0.5793452 0.1431098 0.03648829 0.02745343  
## 99 0.099 0.5793452 0.1431098 0.03648829 0.02745343  
## 100 0.100 0.5793452 0.1431098 0.03648829 0.02745343

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



#Example variable importance in model  
varImp(tree)

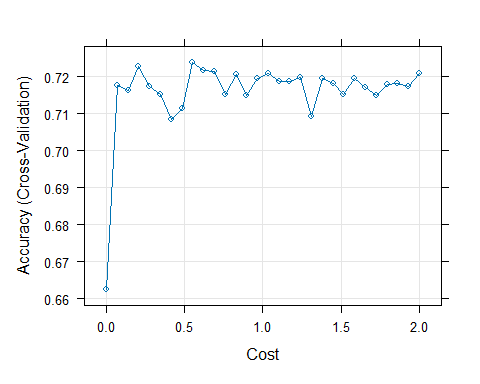
## rpart variable importance  
##   
## only 20 most important variables shown (out of 36)  
##   
## Overall  
## age 100.0000  
## bmi 66.1311  
## educationCollege Grad 26.0499  
## pulse 26.0073  
## hh\_incomemore 99999 19.4667  
## phys\_activeYes 12.6279  
## race1White 4.9343  
## race1Mexican 4.6393  
## smoke100Yes 4.2695  
## hh\_income45000-54999 3.6509  
## hh\_income65000-74999 2.0205  
## hh\_income55000-64999 1.9376  
## race1Hispanic 1.3287  
## hh\_income35000-44999 1.3101  
## educationHigh School 1.1938  
## educationSome College 0.8119  
## hh\_income10000-14999 0.7489  
## hh\_income 5000-9999 0.5465  
## hh\_income15000-19999 0.4811  
## race1Other 0.4501

## 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)



#See information about final model  
svm$finalModel

## Support Vector Machine object of class "ksvm"   
##   
## SV type: C-svc (classification)   
## parameter : cost C = 0.552448275862069   
##   
## Linear (vanilla) kernel function.   
##   
## Number of Support Vectors : 549   
##   
## Objective Function Value : -293.8974   
## Training error : 0.247835   
## 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 62.9 2.4  
## Yes 26.8 8.0  
##   
## Accuracy (average) : 0.7083

confusionMatrix(svm)

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 64.4 2.4  
## Yes 25.2 7.9  
##   
## Accuracy (average) : 0.7238

confusionMatrix(logistic)

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 64.0 2.6  
## Yes 25.7 7.8  
##   
## Accuracy (average) : 0.7175

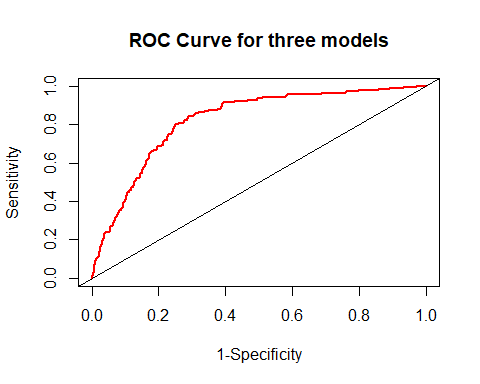
* Since the accuracy for Support Vector Machine is the highest(0.7213), I would chose Support Vector Machine as my optimal model.

## 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 1237 36  
## Yes 472 161  
##   
## Accuracy : 0.7335   
## 95% CI : (0.713, 0.7532)  
## No Information Rate : 0.8966   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.2734   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.81726   
## Specificity : 0.72382   
## Pos Pred Value : 0.25434   
## Neg Pred Value : 0.97172   
## Prevalence : 0.10336   
## Detection Rate : 0.08447   
## Detection Prevalence : 0.33211   
## Balanced Accuracy : 0.77054   
##   
## '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)



# 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.