Predicting the risk of diabetes

STATS/CSE 780 Course Project

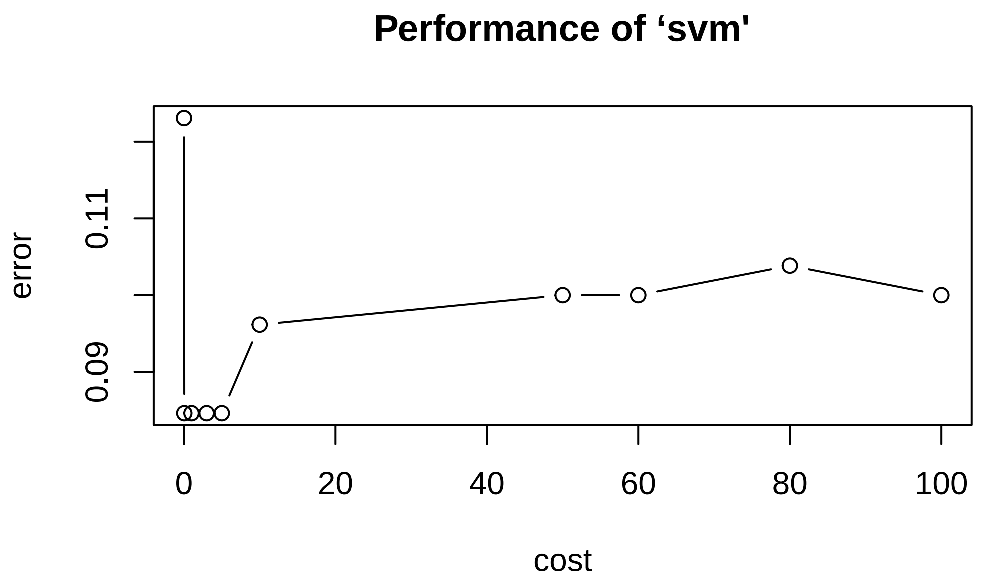
Pao Zhu Vivian Hsu (Student Number: 400547994)

2023-12-12

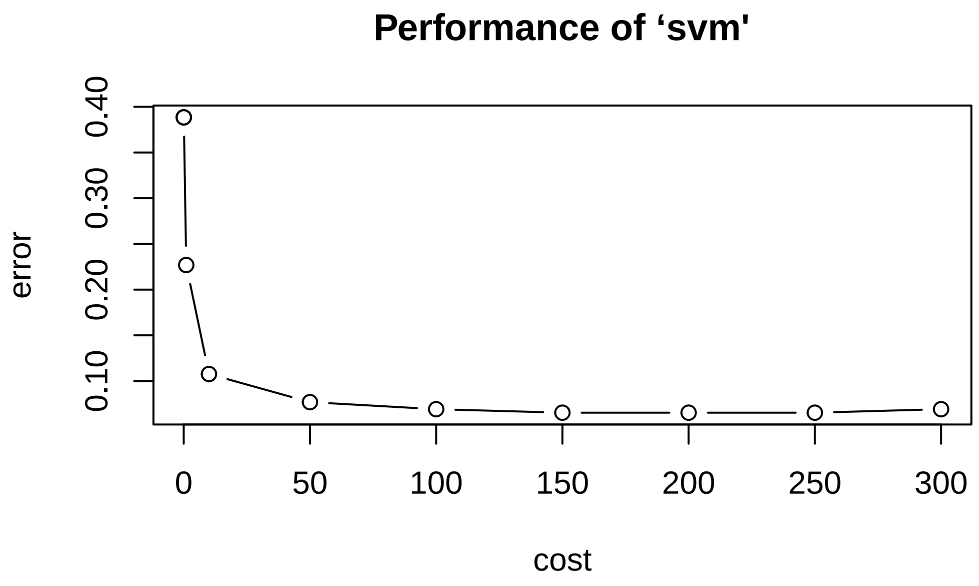
Top of code

```
Call:
svm(formula = class ~ ., data = diabetesTrain_Scaled, kernel = "linear",
    cost = 10, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10

Number of Support Vectors:  55

 ( 24 31 )


Number of Classes:  2

Levels:
 0 1

Parameter tuning of 'svm':
```

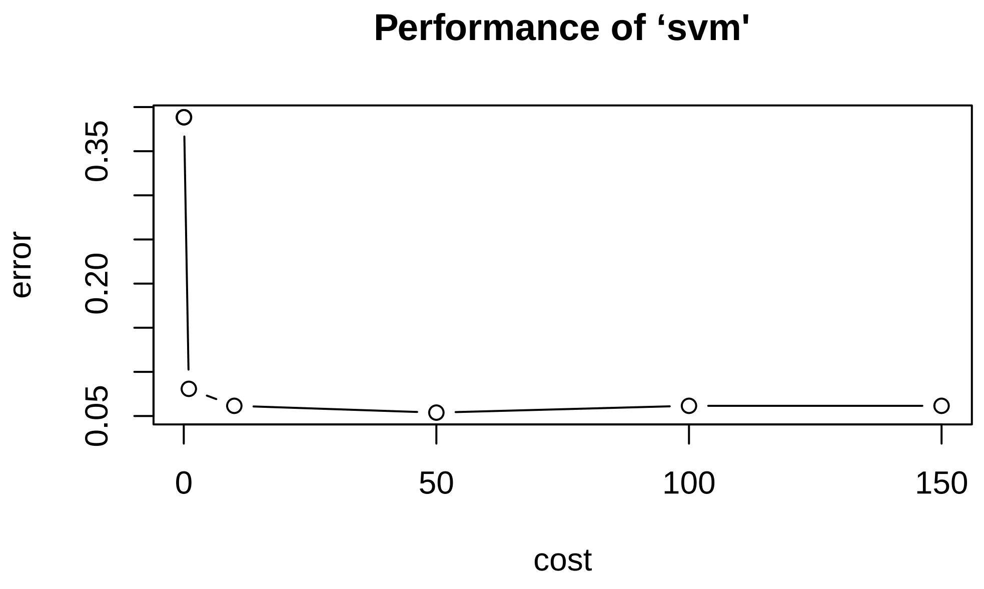
```
- sampling method: 10-fold cross validation


- best parameters:
 cost
    3


- best performance: 0.08461538


- Detailed performance results:
    cost      error dispersion
1  1e-02 0.12307692 0.05378507
2  5e-02 0.08461538 0.05958436
3  1e+00 0.08461538 0.07206907
4  3e+00 0.08461538 0.07861390
5  5e+00 0.08461538 0.07861390
6  1e+01 0.09615385 0.07530346
7  5e+01 0.10000000 0.07734925
8  6e+01 0.10000000 0.07734925
9  8e+01 0.10384615 0.07263700
10 1e+02 0.10000000 0.06832263
```



```

Call:
best.tune(METHOD = svm, train.x = class ~ ., data = diabetesTrain_Scaled,
    ranges = list(cost = c(0.01, 0.05, 1, 3, 5, 10, 50, 60, 80, 100)),
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  3

Number of Support Vectors:  58

 ( 29 29 )


Number of Classes:  2

Levels:
 0 1

       truth
predict   0   1
      0  91   8
      1   8 153

[1] 0.9384615

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation
- best parameters:
 cost
  150

- best performance: 0.06538462

- Detailed performance results:
     cost      error dispersion
1    0.01 0.38846154 0.06892144
2    0.05 0.38846154 0.06892144
3    1.00 0.22692308 0.10641798
4   10.00 0.10769231 0.04727972
5   50.00 0.07692308 0.04796997
6  100.00 0.06923077 0.04366509
7  150.00 0.06538462 0.04074423
8  200.00 0.06538462 0.04074423
9  250.00 0.06538462 0.04814098
10 300.00 0.06923077 0.04366509
```



```
Call:
best.tune(METHOD = svm, train.x = class ~ ., data = diabetesTrain_Scaled,
    ranges = list(cost = c(0.01, 0.05, 1, 10, 50, 100, 150, 200,
        250, 300)), kernel = "polynomial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  150
     degree:  3
     coef.0:  0

Number of Support Vectors:  73

 ( 34 39 )


Number of Classes:  2

Levels:
 0 1

       truth
predict   0   1
      0  99  11
      1   0 150

[1] 0.9576923

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation
- best parameters:
 cost
   50

- best performance: 0.05384615

- Detailed performance results:
    cost      error dispersion
1   0.01 0.38846154 0.06892144
2   0.05 0.38846154 0.06892144
3   1.00 0.08076923 0.05573606
4  10.00 0.06153846 0.04514568
5  50.00 0.05384615 0.04514568
6 100.00 0.06153846 0.04134491
7 150.00 0.06153846 0.04134491
```



```
Call:
best.tune(METHOD = svm, train.x = class ~ ., data = diabetesTrain_Scaled,
    ranges = list(cost = c(0.01, 0.05, 1, 10, 50, 100, 150)), kernel = "radial")
```

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  50

Number of Support Vectors:  56

 ( 23 33 )


Number of Classes:  2

Levels:
 0 1

       truth
predict   0   1
      0  98   5
      1   1 156

[1] 0.9769231
```

Abstract

Introduction

Diabetes is a chronic disease that occurs when the body cannot effectively produce or use insulin to regulate sugar levels in the blood. According to the World Health Organization, 422 million people have diabetes worldwide and 1.5 million deaths that occur every year are directly linked to the disease (World Health Organization, n.d.). Due to its high prevalence and burden, finding ways to predict diabetes has become critical in improving population health.

With today's technological capabilities and abundance of data, machine learning has become the forefront of diabetes prediction. In literature, a large variety of techniques, such as naive Bayes, decision trees, and neural networks have been studied to find the most accurate method of prediction (Kumar & Velide, 2014; Rabina & Chopra, 2016).

One notable example is Islam et al.'s study on diabetes prediction (2020). In this study, the authors collected data from patients in Sylhet Diabetes Hospital in Sylhet, Bangladesh. Naive Bayes, logistic regression, and random forest techniques were applied to the data to predict the risk of diabetes. The study concluded that a random forest model had the greatest accuracy (Islam et al., 2020).

In our study, we also utilize machine learning techniques to predict the risk of diabetes. In particular, we focus on decision tree and neural network methods. The data we used was downloaded from an open source website called Kaggle (Larxel, 2023) and is the same data used in Islam et al.'s study (2020).

Methods

Data Exploration

The data set consists of 520 observations and 17 attributes. Before any analysis was done, a transformation was applied to the data to ensure that all categorical variables were expressed with binary indicators to ease the analysis. The response variable is a binary attribute called class that indicates whether the patient has a positive or negative risk for diabetes. The remaining attributes describe the patient and if they experience common symptoms related to the disease, such as weakness, itching, and obesity. A full list of the attributes and their meanings are outlined in Supp. Table 1.

There are no missing values in this data set since missing data was already addressed by Islam et al. after data collection (2020). To verify this, we performed a check for nulls and blanks as summarized in Supp. Table 2.

A correlation plot was created to check if there are any strong correlations between the attributes. We define a strong correlation as those with a correlation coefficient of 0.7 or larger. Based on Figure 1, all values are lower than 0.7 so there is no evidence of strong correlations.

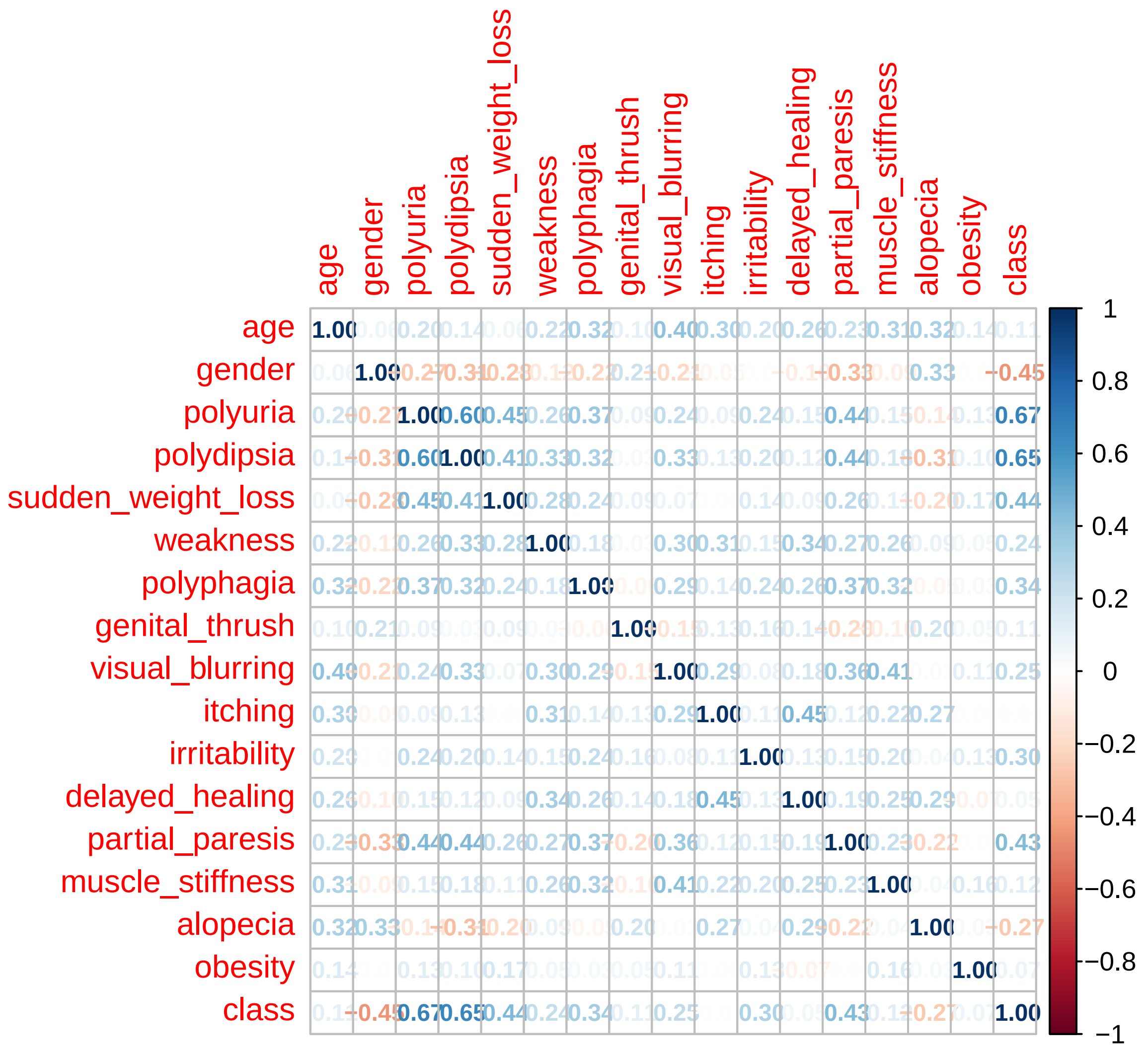


Figure 1: Correlation plot

Next, each of the individual attributes were explored. Figure 2 below shows a box plot of patient ages. The median age in the population is 47. There are a few outliers that exist outside of the interquartile range. These values will be capped at 79, the upper bound of the range.

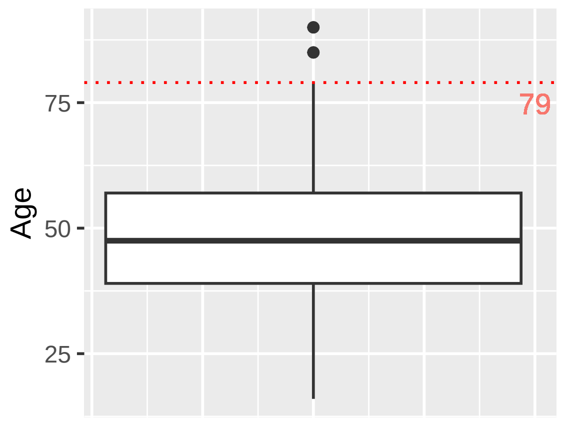


Figure 2: Boxplot of age

We also plotted the 16 categorical variables in bar charts as illustrated in Supp. Figure 3. There appears to be a somewhat even split between the predictor values for polyuria and itching, while the remaining predictors are not evenly split. This is especially important for the class variable because there are about 200 observations with a negative diabetes risk and about 300 observations with a positive risk. This suggests that if the model does not perform well, a resampling method such as cross-validation or bootstrapping may help improve the model.

Decision Tree

The first machine learning method will be a decision tree. This method was selected because Islam et al.'s study found that a decision tree model was most accurate (2020) and so this study will aim to reproduce such results. The index on the subtrees α will be tuned using cross-validation. Pruning will be applied to reduce the cost complexity of the tree, and random forest ensembling will be used to improve model performance.

The first method

Support Vector Machine

The second method will be a support vector machine (SVM). This method was selected since . A neural network method was considered, but we noticed there are only 520 observations. Neural networks are optimal for large datasets while SVM performs well with smaller datasets and especially for classification problems.

Neural Network

The second method will be a neural network. This method was selected since Islam et al.'s study has not explored a model using neural networks (2020) and so there is a possibility that a neural network may be more accurate than a tree-based model. The number of hidden layers and the units per layer will be tuned through trial and error. Based on James et al.'s book on statistical learning (2021), the units per layer will be set to some large value and overfitting will be controlled with ridge regularization. The strength of the regularization λ will be tuned at each layer.

Results

After the two models are built, they will each be assessed using misclassification rate, accuracy, specificity, and sensitivity. A comparison of these four measurements will reveal which model is a stronger fit to predict diabetes. By nature, decision trees are easier to interpret and reproducible compared to neural networks. Thus, these qualities will also be considered in its comparison.

Supplementary Materials

Tables and Figures

Table 1: Description of attributes

Attribute	Values
Age	In years
Gender	1 = Male, 0 = Female
Polyuria	1 = Yes, 0 = No
Polydipsia	1 = Yes, 0 = No
Sudden weight loss	1 = Yes, 0 = No
Weakness	1 = Yes, 0 = No
Polyphagia	1 = Yes, 0 = No
Genital thrush	1 = Yes, 0 = No
Visual blurring	1 = Yes, 0 = No
Itching	1 = Yes, 0 = No
Irritability	1 = Yes, 0 = No
Delayed healing	1 = Yes, 0 = No
Partial paresis	1 = Yes, 0 = No
Muscle stiffness	1 = Yes, 0 = No
Alopecia	1 = Yes, 0 = No
Obesity	1 = Yes, 0 = No
Class	1 = Positive risk, 0 = Negative risk

Table 2: No missing data

Nulls	Blanks
0	0

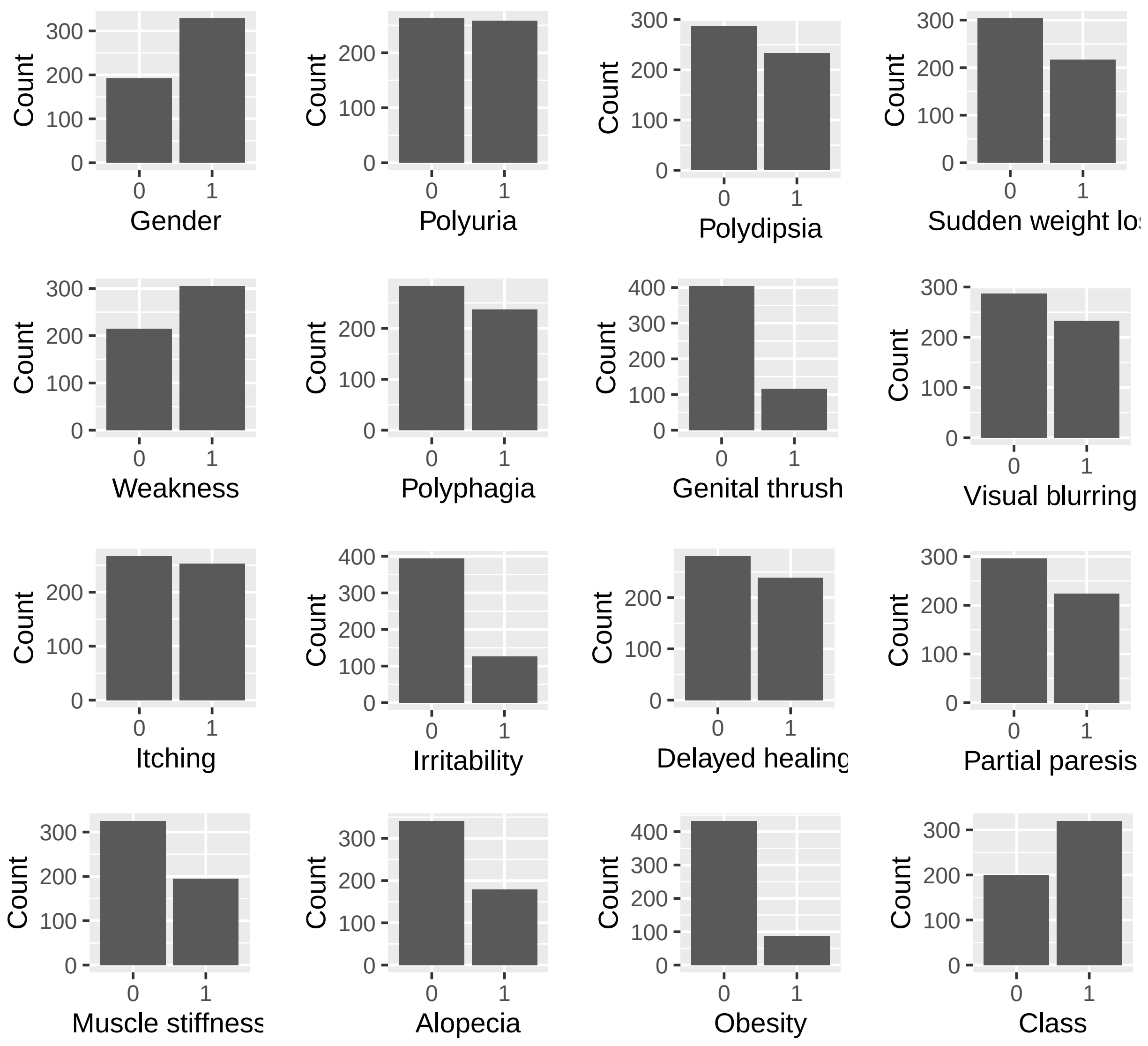


Figure 3: Bar charts of categorical variables

Code

```
library(knitr)

library(tidyverse)

library(corrplot)

library(tree)

library(randomForest)

library(e1071)
```

```r
# ----- DATA CLEANSING ----- #

diabetes_raw <- read.csv("diabetes_data.csv", sep=";")

diabetes_int <- diabetes_raw %>%

  mutate(gender = as.integer(ifelse(gender=="Male",1,

                          ifelse(gender=="Female",0,

                          NA))))

diabetes_pretransform <- diabetes_raw %>%

  mutate(gender = as.factor(ifelse(gender=="Male",1,

                          ifelse(gender=="Female",0,

                          NA))),

         polyuria = as.factor(polyuria),

         polydipsia = as.factor(polydipsia),

         sudden_weight_loss = as.factor(sudden_weight_loss),

         weakness = as.factor(weakness),

         polyphagia = as.factor(polyphagia),

         genital_thrush = as.factor(genital_thrush),

         visual_blurring = as.factor(visual_blurring),

         itching = as.factor(itching),

         irritability = as.factor(irritability),

         delayed_healing = as.factor(delayed_healing),

         partial_paresis = as.factor(partial_paresis),

         muscle_stiffness = as.factor(muscle_stiffness),

         alopecia = as.factor(alopecia),

         obesity = as.factor(obesity),

         class = as.factor(class))


# ----- DATA EXPLORATION ----- #

# Data description

attribute <- c("Age","Gender","Polyuria","Polydipsia",

               "Sudden weight loss","Weakness","Polyphagia",

               "Genital thrush","Visual blurring","Itching",
               "Irritability","Delayed healing","Partial paresis",
               "Muscle stiffness","Alopecia","Obesity","Class")
values <- c("In years","1 = Male, 0 = Female","1 = Yes, 0 = No","1 = Yes, 0 = No",
            "1 = Yes, 0 = No","1 = Yes, 0 = No","1 = Yes, 0 = No","1 = Yes, 0 = No",
            "1 = Yes, 0 = No","1 = Yes, 0 = No","1 = Yes, 0 = No","1 = Yes, 0 = No",
            "1 = Yes, 0 = No","1 = Yes, 0 = No","1 = Yes, 0 = No","1 = Yes, 0 = No",
            "1 = Positive risk, 0 = Negative risk")
data_summary <- data.frame(Attribute=attribute, Values=values)
kable(data_summary)

# Check for missing data
nulls <- sapply(diabetes_pretransform,
                function(col){ifelse(is.na(sum(col == "")), 0, sum(col == ""))})
blanks <- sapply(diabetes_pretransform,
                 function(col){ifelse(is.na(sum(col == "")), 0, sum(col == ""))})
kable(data.frame(Nulls=sum(nulls), Blanks=sum(blanks)))

# Boxplot of continuous variable
bplot <- ggplot(diabetes_pretransform, aes(y = age)) + geom_boxplot() + labs(x="",y="Age") +
  theme(axis.text.x=element_blank(), axis.ticks.x=element_blank())
bplot_a1 <- as.integer(unlist(ggplot_build(bplot)$data)["ymax"])
bplot + geom_hline(yintercept = bplot_a1, linetype="dotted", color="red") +
  geom_text(aes(0.4,bplot_a1,label = bplot_a1, vjust = 1.5, color="red"),
            show.legend = FALSE)

# Cap outliers
diabetes <- diabetes_pretransform %>% mutate(age = ifelse(age > bplot_a1, bplot_a1, age))

# Bar charts for each categorical variable
ggplot(diabetes_pretransform, aes(x = gender)) + geom_bar() + labs(y = "Count", x = "Gender"
ggplot(diabetes_pretransform, aes(x = polyuria)) + geom_bar() + labs(y = "Count", x = "Polyu
ggplot(diabetes_pretransform, aes(x = polydipsia)) + geom_bar() +
  labs(y = "Count", x = "Polydipsia")
ggplot(diabetes_pretransform, aes(x = sudden_weight_loss)) + geom_bar() +
  labs(y = "Count", x = "Sudden weight loss")
ggplot(diabetes_pretransform, aes(x = weakness)) + geom_bar() + labs(y = "Count", x = "Weakn
ggplot(diabetes_pretransform, aes(x = polyphagia)) + geom_bar() +
  labs(y = "Count", x = "Polyphagia")
ggplot(diabetes_pretransform, aes(x = genital_thrush)) + geom_bar() +
  labs(y = "Count", x = "Genital thrush")
ggplot(diabetes_pretransform, aes(x = visual_blurring)) + geom_bar() +
  labs(y = "Count", x = "Visual blurring")
ggplot(diabetes_pretransform, aes(x = itching)) + geom_bar() + labs(y = "Count", x = "Itchin
ggplot(diabetes_pretransform, aes(x = irritability)) + geom_bar() +
  labs(y = "Count", x = "Irritability")
ggplot(diabetes_pretransform, aes(x = delayed_healing)) + geom_bar() +
  labs(y = "Count", x = "Delayed healing")
ggplot(diabetes_pretransform, aes(x = partial_paresis)) + geom_bar() +
  labs(y = "Count", x = "Partial paresis")
ggplot(diabetes_pretransform, aes(x = muscle_stiffness)) + geom_bar() +
  labs(y = "Count", x = "Muscle stiffness")
ggplot(diabetes_pretransform, aes(x = alopecia)) + geom_bar() + labs(y = "Count", x = "Alope
ggplot(diabetes_pretransform, aes(x = obesity)) + geom_bar() + labs(y = "Count", x = "Obesit
ggplot(diabetes_pretransform, aes(x = class)) + geom_bar() + labs(y = "Count", x = "Class")

# Correlation plot
corr_matrix <- cor(diabetes_int)
corrplot(round(corr_matrix,2), method = "number", number.cex=0.75)
# ----- DATA SPLITTING ----- #
# Split the data into test and training set
set.seed(2023)
trainIndex <- sample(1:nrow(diabetes), round(nrow(diabetes)/2, 0), replace = FALSE)
```

```r
diabetesTrain <- diabetes[trainIndex, ]
diabetesTest <- diabetes[-trainIndex, ]

# Pull out classes for testing
diabetesTest_class <- diabetes$class[-trainIndex]

# ----- DECISION TREE ----- #
# Fit the classification tree
treeDiabetes <- tree(
  class ~ .,
  data = diabetes,
  subset = trainIndex,
  split = "deviance")

# Summary of tree results
summary(treeDiabetes)
table(diabetes$class)/sum(table(diabetes$class))

# Graphical summary of fitted tree
plot(treeDiabetes)
text(treeDiabetes, pretty = 0) #include the category names for any qualitative predictors

# Explore the fitted tree
print(treeDiabetes)
# Use the fitted tree to predict results for the test data
treeDiabetesPred <- predict(treeDiabetes, diabetesTest, type = "class")
head(treeDiabetesPred)
table(treeDiabetesPred, diabetesTest_class)

# Percent that was correctly predicted
(table(treeDiabetesPred, diabetesTest_class)[1,1] +
    table(treeDiabetesPred, diabetesTest_class)[2,2])/sum(table(treeDiabetesPred, diabetesTe
# Get misclassification rate of different tree sizes to use for pruning
set.seed(2023)
diabetesCV <- cv.tree(treeDiabetes, FUN = prune.misclass)
names(diabetesCV)
diabetesCV

plot(diabetesCV$size, diabetesCV$dev, type = "b")
plot(diabetesCV$k, diabetesCV$dev, type = "b")
# Prune the tree using the size that produces the lowest misclassification rate
treeDiabetesPrune <- prune.misclass(treeDiabetes, best = 12)
plot(treeDiabetesPrune)
text(treeDiabetesPrune, pretty = 0)
# Use the fitted tree to predict results for the test data
treeDiabetesPrunePred <- predict(treeDiabetesPrune, diabetesTest, type = "class")
head(treeDiabetesPrunePred)
table(treeDiabetesPrunePred, diabetesTest_class)

# Percent that was correctly predicted
(table(treeDiabetesPrunePred, diabetesTest_class)[1,1] +
    table(treeDiabetesPrunePred, diabetesTest_class)[2,2])/sum(table(treeDiabetesPrunePred,
# Random forest to improve results
set.seed(2023)
diabetesRF <- randomForest(
  class ~ ., data = diabetes,
  subset = trainIndex,
  mtry = 4,
  importance = TRUE
  )
yhatRF <- predict(
  diabetesRF, newdata = diabetesTest
  )

# Percent that was correctly predicted
(table(yhatRF, diabetesTest_class)[1,1] +
    table(yhatRF, diabetesTest_class)[2,2])/sum(table(yhatRF, diabetesTest_class))
# Importance of the variables
varImpPlot(diabetesRF)

# ----- SUPPORT VECTOR MACHINE ----- #
# Scale the data so the values are from 0 to 1
normalize0to1 <- function(data){
  (data-min(data))/(max(data)-min(data))
}
diabetesTrain_Scaled <- diabetesTrain
diabetesTest_Scaled <- diabetesTest
diabetesTrain_Scaled$age <- normalize0to1(diabetesTrain$age)
diabetesTest_Scaled$age <- normalize0to1(diabetesTest$age)

# Fit an SVM model
set.seed(2023)
svmfit <- svm(
  class ~ .,
  data = diabetesTrain_Scaled,
  kernel = "linear",
  cost = 10,
  scale = FALSE
  )

# Predict and compute accuracy
ypred <- predict(svmfit, diabetesTest_Scaled)
table(predict = ypred, truth = diabetesTest_Scaled$class)
```

```r
(table(ypred, diabetesTest_class)[1,1] +
    table(ypred, diabetesTest_class)[2,2])/sum(table(ypred, diabetesTest_class))
summary(svmfit)
# Use a linear kernel and perform cross validation to find the best cost
set.seed(2023)
tune.out <- tune(
  svm,
  class ~ .,
  data = diabetesTrain_Scaled,
  kernel = "linear",
  ranges = list(
    cost = c(0.01, 0.05, 1, 3, 5, 10, 50, 60, 80, 100)
    )
  )
summary(tune.out)
plot(tune.out)
bestmod <- tune.out$best.model
summary(bestmod)

# Predict and compute accuracy
ypred <- predict(bestmod, diabetesTest_Scaled)
table(predict = ypred, truth = diabetesTest_Scaled$class)

(table(ypred, diabetesTest_class)[1,1] +
    table(ypred, diabetesTest_class)[2,2])/sum(table(ypred, diabetesTest_class))
# Use a polynomial kernel and perform cross validation to find the best cost
set.seed(2023)
tune.out <- tune(
  svm,
  class ~ .,
  data = diabetesTrain_Scaled,
  kernel = "polynomial",
  ranges = list(
    cost = c(0.01, 0.05, 1, 10, 50, 100, 150, 200, 250, 300)
    )
  )
summary(tune.out)
plot(tune.out)
bestmod <- tune.out$best.model
summary(bestmod)

# Predict and compute accuracy
ypred <- predict(bestmod, diabetesTest_Scaled)
table(predict = ypred, truth = diabetesTest_Scaled$class)

(table(ypred, diabetesTest_class)[1,1] +
    table(ypred, diabetesTest_class)[2,2])/sum(table(ypred, diabetesTest_class))
# Use a polynomial kernel and perform cross validation to find the best cost
set.seed(2023)
tune.out <- tune(
  svm,
  class ~ .,
  data = diabetesTrain_Scaled,
  kernel = "radial",
  ranges = list(
    cost = c(0.01, 0.05, 1, 10, 50, 100, 150)
    )
  )
summary(tune.out)
plot(tune.out)
bestmod <- tune.out$best.model
```

```r
summary(bestmod)


# Predict and compute accuracy
ypred <- predict(bestmod, diabetesTest_Scaled)
table(predict = ypred, truth = diabetesTest_Scaled$class)


(table(ypred, diabetesTest_class)[1,1] +
    table(ypred, diabetesTest_class)[2,2])/sum(table(ypred, diabetesTest_class))
```

References

Islam, M. M. F., Ferdousi, R., Rahman, S., & Bushra, H. Y. (2020). Likelihood prediction of diabetes at early stage using data mining techniques. In M. Gupta, D. Konar, S. Bhattacharyya, & S. Biswas (Eds.), *Computer vision and machine intelligence in medical image analysis* (pp. 113–125). Springer Singapore. https://doi.org/10.1007/978-981-13-8798-2_12

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). Deep learning. In *An introduction to statistical learning with applications in r 2nd edition* (pp. 403–458). Springer.

Kumar, V., & Velide, L. (2014). *A data mining approach for prediction and treatment of diabetes disease.*

Larxel. (2023). *Early classification of diabetes.* https://www.kaggle.com/datasets/andrewmvd/heart-failure-clinical-data/data

R Core Team. (2023). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing. https://www.R-project.org/

Rabina, & Chopra, Er. A. (2016). *Diabetes prediction by supervised and unsupervised learning with feature selection.*

World Health Organization. (n.d.). *Diabetes.* https://www.who.int/health-topics/diabetes#tab=tab_1