Predicting the risk of diabetes

STATS/CSE 780 Course Project

Pao Zhu Vivian Hsu (Student Number: 400547994)

2023-12-12

Top of code

Decision tree

SVM

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 (2023). Due to its large impact, finding a way to accurately predict diabetes has become paramount to improve global health. In this paper, we aim to address this problem by leveraging machine learning techniques to predict the risk of diabetes.

Various machine learning techniques have already been studied in current literature to find an accurate model to predict the disease (Kumar & Velide, 2014; Rabina & Chopra, 2016). One notable example is Islam et al.'s study which analyzes symptoms from patients of a diabetes hospital in Sylhet, Bangladesh (2020). The authors used Naive Bayes, logistic regression, and random forest methods, and found that their random forest model had the greatest accuracy among the three methods used (Islam et al., 2020).

In our study, we perform a similar analysis using decision tree and support vector machine (SVM) methods. The data we have used was downloaded from an open source website called Kaggle (Larxel, 2023) and is the same data in Islam et al.'s study (2020). This allows us to compare our results with theirs in extension of their research.

Methods

To begin the study, we performed data visualization and handled any detected problems by applying data transformation. We then randomly split the data into two equal parts to use as training and testing sets for both of the methods.

The first method we used was 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 our goal is to validate and reproduce the results of their tree. The first tree we fit had a terminal size of 16 nodes. To improve the accuracy and reduce the cost complexity of the tree, we used cross-validation as shown in Figure 1 below and found that a size of 12 nodes produced the lowest misclassification error rate. Pruning was applied using this node size. After pruning the tree, we then used random forest ensembling to further improve model performance.

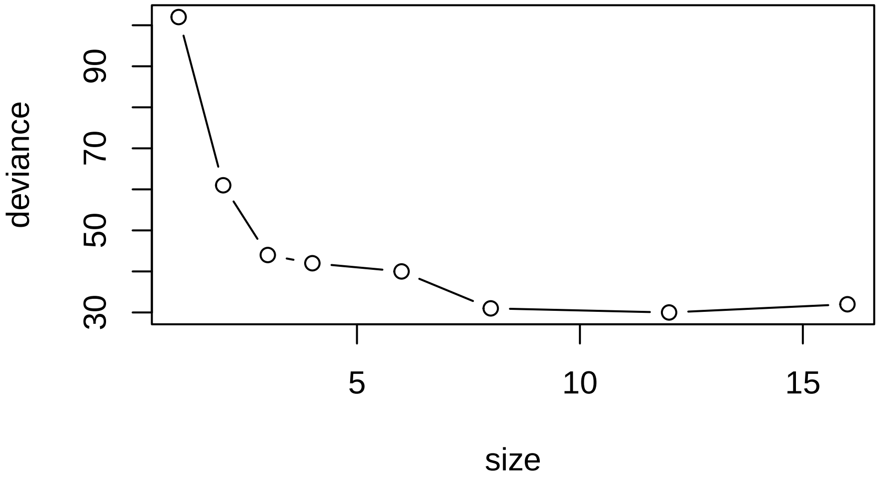

Figure 1: Cross-validation of decision tree prior to pruning; Optimal size = 12

The second method we used was SVM. This method was selected because Islam et al. (2020) have not used this method in their analyses, and so our goal is to determine whether an SVM model would better predict the risk of diabetes compared to a tree-based model. In addition, SVMs are considered as a strong model choice for binary classification (James et al., 2013), which is the type of data problem we are working with. We first started by scaling the data to ensure that units are between 0 and 1 across all variables. Since majority of the variables are categorical and one-hot encoded, we only needed to scale the age variable. We then performed SVM multiple times using kernel adjustments and cross-validation for cost. The three kernel types we investigated include linear, polynomial, and radial basis function (RBF) kernels. For each of these kernels, we used cross-validation to determine the cost that would produce the lowest misclassification error rate. As illustrated in Figure 2, Figure 3, and Figure 4, the optimal costs were 3, 150, and 50 for the linear, polynomial, and RBF kernels respectively.

After building each model, we computed the accuracy, specificity, sensitivity, and misclassification error rate to assess model performance. These performance parameters were selected because they are often used for binary classification problems. Moreover, a few of these metrics were used by Islam et al. (2020), so computing them facilitated direct comparison with their models. A comparison of these four measurements between models helped us determine which model is a stronger fit to predict diabetes.

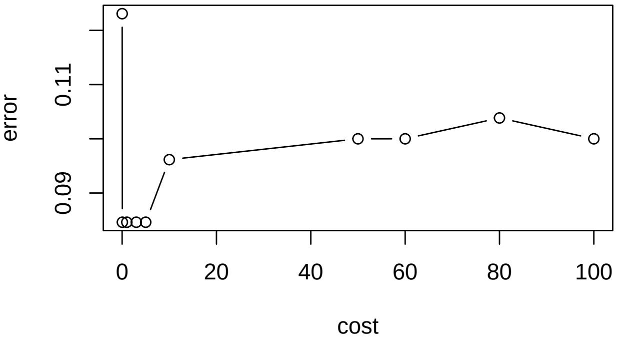

Figure 2: Cross-validation for linear SVM; Optimal cost = 3

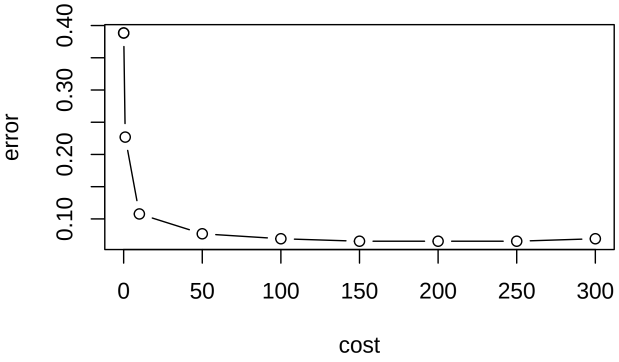

Figure 3: Cross-validation for polynomial SVM; Optimal cost = 150

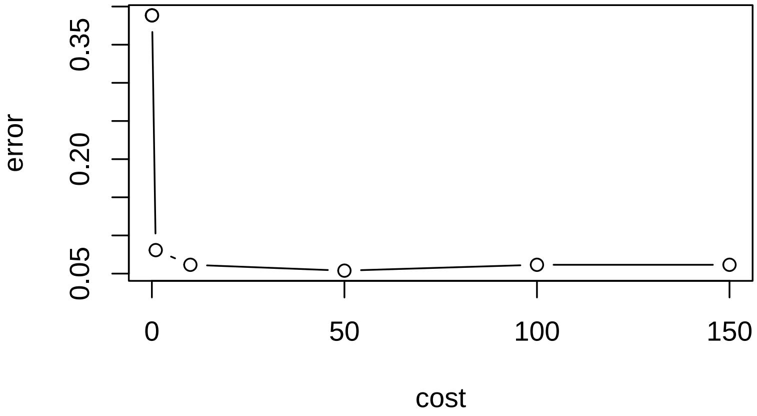

Figure 4: Cross-validation for RBF SVM; Optimal cost = 50

Results

Exploratory Data Analysis

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 4.

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 5.

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 5, all values are lower than 0.7 so there is no evidence of strong correlations.

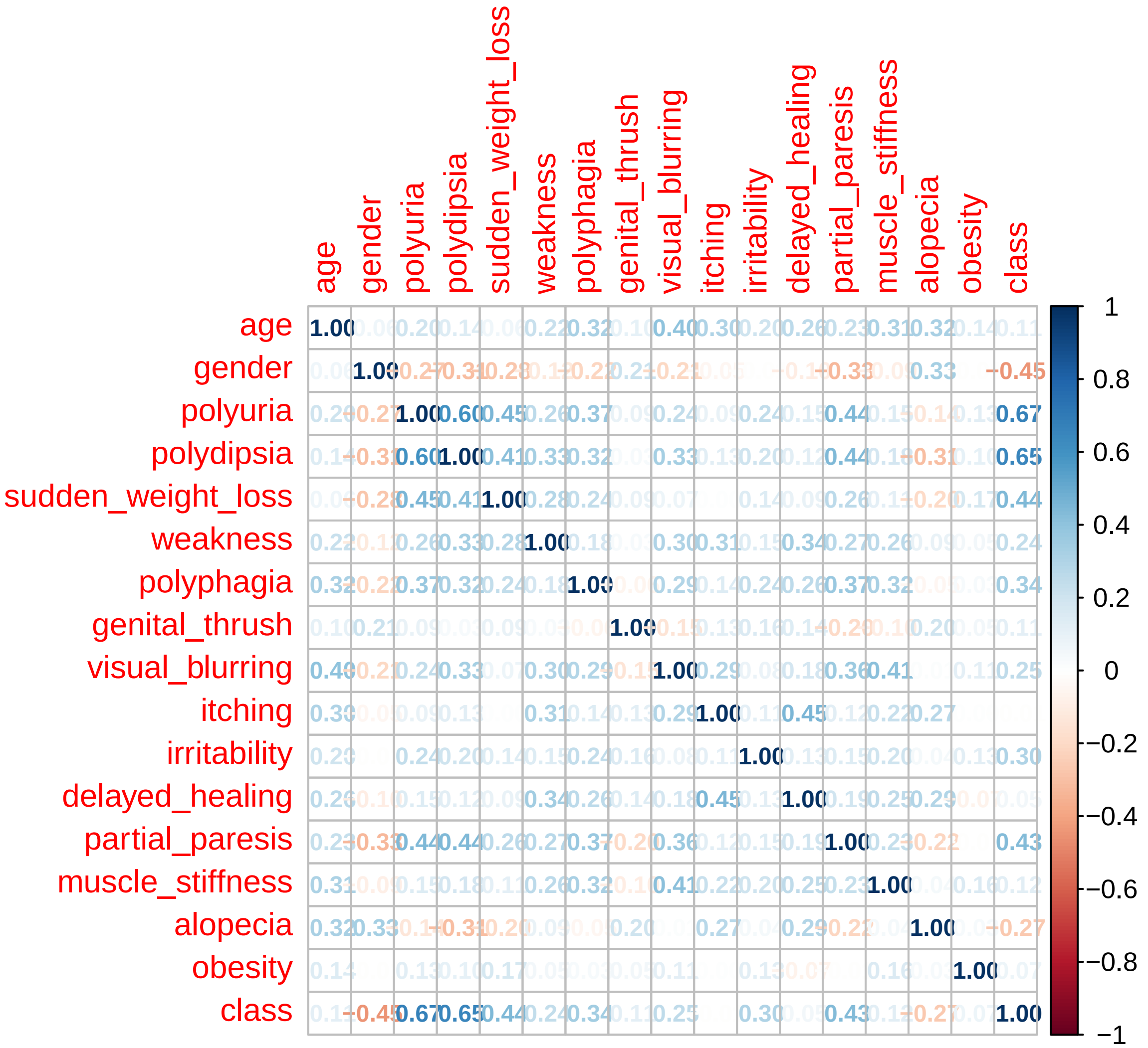

Figure 5: Correlation plot

Next, each of the individual attributes were explored. Figure 6 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 were capped at 79, the upper bound of the range.

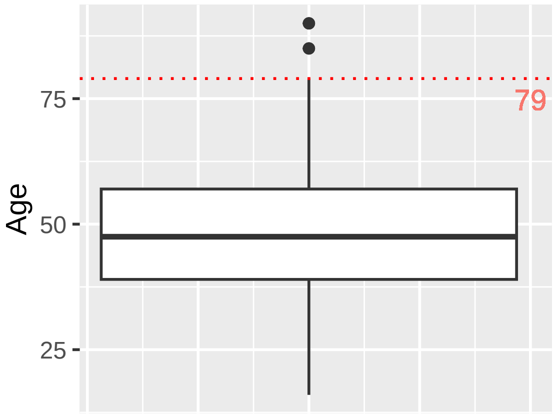

Figure 6: Boxplot of age

We also plotted the 16 categorical variables in bar charts as illustrated in Supp. Figure 9. 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. As such, we have first modelled the data in its original distribution and made adjustments as required.

Decision Tree

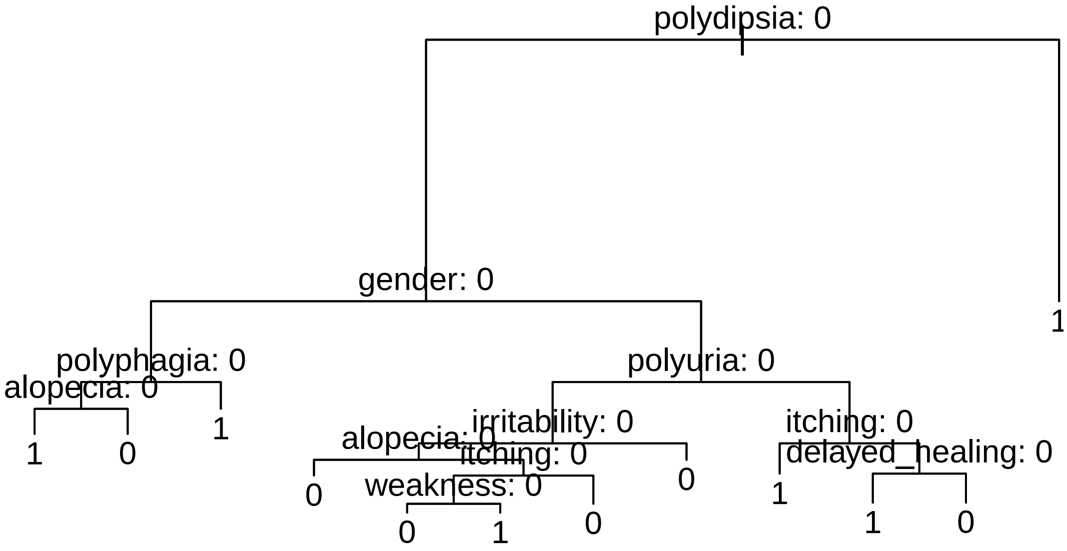

Figure 7: Pruned decision tree with 12 terminal nodes

Table 1: Model performance parameters for tree-based methods

Model	Accuracy	Sensitivity	Specificity	Misclassification Rate
Initial tree, 16 nodes	94.615%	95.652%	92.929%	5.385%
Pruned tree, 12 nodes	94.615%	95.652%	92.929%	5.385%
Random forest	98.846%	98.765%	98.98%	1.154%

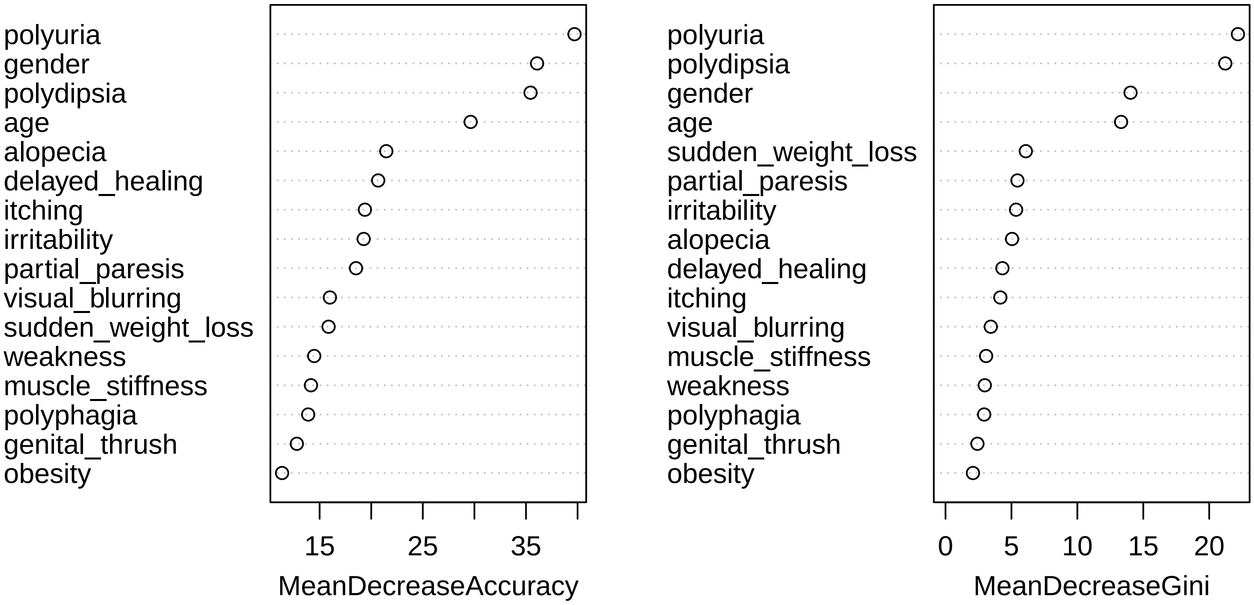

Figure 8: Importance of variables from random forest

Support Vector Machine

Table 2: Model performance parameters for SVM methods

Model	Accuracy	Sensitivity	Specificity	Misclassification Rate
Linear	93.846%	95.031%	91.919%	6.154%
Polynomial	95.769%	100%	90%	4.231%
Radial	97.692%	99.363%	95.146%	2.308%

Conclusion

Table 3: Comparison of performance parameters between best tree-based and SVM model

Model	Accuracy	Sensitivity	Specificity	Misclassification Rate
Random forest	98.846%	98.765%	98.98%	1.154%
SVM with RBF kernel	97.692%	99.363%	95.146%	2.308%

Supplementary Materials

Tables and Figures

Table 4: 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 5: No missing data

Nulls	Blanks
0	0

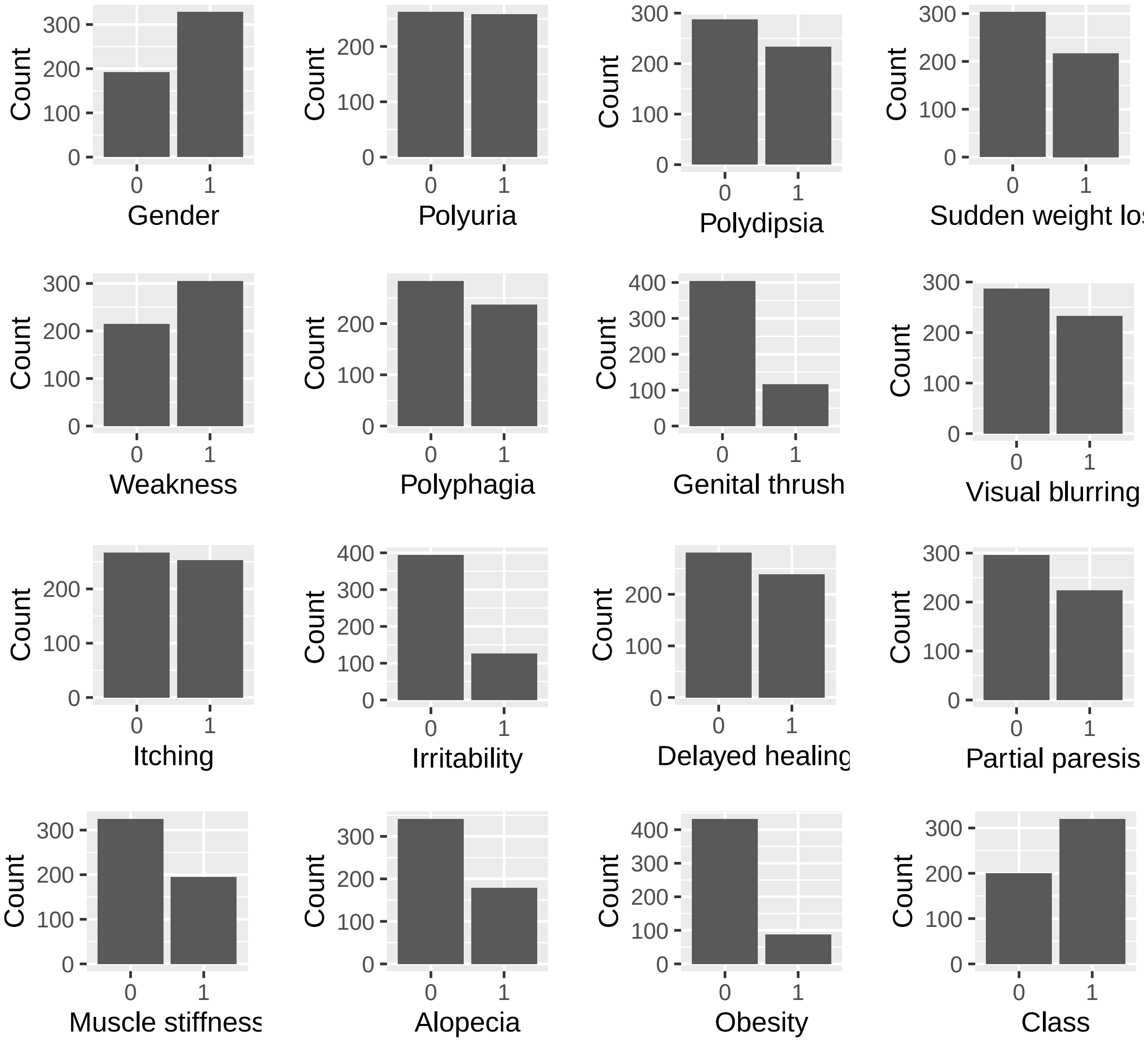

Figure 9: Bar charts of categorical variables

Code

```
library(knitr)

library(tidyverse)

library(corrplot)

library(tree)

library(randomForest)

library(e1071)
```

```
# ----- 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 using upper adjacent value
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 = "Polyuria")
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 = "Weakness")
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 = "Itching")
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 = "Alopecia")
ggplot(diabetes_pretransform, aes(x = obesity)) + geom_bar() +
  labs(y = "Count", x = "Obesity")
ggplot(diabetes_pretransform, aes(x = class)) + geom_bar() +
  labs(y = "Count", x = "Class")
```

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

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

# ----- DECISION TREE ----- #
# Fit the classification tree
dTree <- tree(
  class ~ .,
  data = diabetes,
  subset = trainIndex,
  split = "deviance")
plot(dTree)
text(dTree, pretty = 0)

# Use the fitted tree to predict results for the test data
dTree_pred <- predict(dTree, diabetesTest, type = "class")

# Compute accuracy, sensitivity, and specificity
dTree_cmat <- table(dTree_pred, diabetesTest_class)
dTree_accuracy <- (dTree_cmat[1,1] + dTree_cmat[2,2])/sum(dTree_cmat)
```

```r
dTree_sensitivity <- dTree_cmat[2,2]/sum(dTree_cmat[2,])
dTree_specificity <- dTree_cmat[1,1]/sum(dTree_cmat[1,])
dTree_misclass <- 1-dTree_accuracy

# Get misclassification rate of different tree sizes to use for pruning
set.seed(2023)
dTreeCV <- cv.tree(dTree, FUN = prune.misclass)

# Find tree size that produces lowest misclassification rate
best_dev <- min(dTreeCV$dev)
best_size <- dTreeCV$size[dTreeCV$dev == best_dev]

# Plot the cross-validation on a chart
plot(dTreeCV$size, dTreeCV$dev, type = "b", xlab="size", ylab="deviance")

# Prune the tree using the size with the lowest misclassification rate
dPruneTree <- prune.misclass(dTree, best = best_size)
plot(dPruneTree)
text(dPruneTree, pretty = 0)

# Use the fitted tree to predict results for the test data
dPruneTree_pred <- predict(dPruneTree, diabetesTest, type = "class")

# Compute accuracy, sensitivity, and specificity
dPruneTree_cmat <- table(dPruneTree_pred, diabetesTest_class)
dPruneTree_accuracy <- (dPruneTree_cmat[1,1] + dPruneTree_cmat[2,2])/
                        sum(dPruneTree_cmat)
dPruneTree_sensitivity <- dPruneTree_cmat[2,2]/sum(dPruneTree_cmat[2,])
dPruneTree_specificity <- dPruneTree_cmat[1,1]/sum(dPruneTree_cmat[1,])
dPruneTree_misclass <- 1-dPruneTree_accuracy
```

```r
# Random forest to improve results
set.seed(2023)
dForest <- randomForest(
  class ~ ., data = diabetes,
  subset = trainIndex,
  mtry = 4,
  importance = TRUE)

# Use the random forest to predict results for the test data
dForest_pred <- predict(dForest, newdata = diabetesTest)

# Compute accuracy, sensitivity, and specificity
dForest_cmat <- table(dForest_pred, diabetesTest_class)
dForest_accuracy <- (dForest_cmat[1,1] + dForest_cmat[2,2])/sum(dForest_cmat)
dForest_sensitivity <- dForest_cmat[2,2]/sum(dForest_cmat[2,])
dForest_specificity <- dForest_cmat[1,1]/sum(dForest_cmat[1,])
dForest_misclass <- 1-dForest_accuracy

# Importance of the variables
varImpPlot(dForest, main="")

# Summary table of stats
as.3percent <- function(value){paste0(round(value*100,3),"%")}
model <- c("Initial tree, 16 nodes", "Pruned tree, 12 nodes", "Random forest")
accuracy <- c(dTree_accuracy, dPruneTree_accuracy, dForest_accuracy)
sensitivity <- c(dTree_sensitivity, dPruneTree_sensitivity, dForest_sensitivity)
specificity <- c(dTree_specificity, dPruneTree_specificity, dForest_specificity)
misclass <- c(dTree_misclass, dPruneTree_misclass, dForest_misclass)
tree_summary <- data.frame("Model" = model,
                           "Accuracy" = as.3percent(accuracy),
                           "Sensitivity" = as.3percent(sensitivity),
                           "Specificity" = as.3percent(specificity),
                           "Misclassification Rate" = as.3percent(misclass),
                           check.names = FALSE)
kable(tree_summary)
# ----- 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)

# Use a linear kernel and perform cross validation to find the best cost
set.seed(2023)
tune_linear <- tune(
  svm,
  class ~ .,
  data = diabetesTrain_Scaled,
  kernel = "linear",
  ranges = list(cost = c(0.01, 0.05, 1, 3, 5, 10, 50, 60, 80, 100)))

plot(tune_linear, main = "")
bestmod_linear <- tune_linear$best.model

# Use the linear svm to predict results for the test data
linear_pred <- predict(bestmod_linear, diabetesTest_Scaled)

# Compute accuracy, sensitivity, and specificity
linear_cmat <- table(linear_pred, diabetesTest_Scaled$class)
```

```r
linear_accuracy <- (linear_cmat[1,1] + linear_cmat[2,2])/sum(linear_cmat)
linear_sensitivity <- linear_cmat[2,2]/sum(linear_cmat[2,])
linear_specificity <- linear_cmat[1,1]/sum(linear_cmat[1,])
linear_misclass <- 1-linear_accuracy

# Use a polynomial kernel and perform cross validation to find the best cost
set.seed(2023)
tune_poly <- tune(
  svm,
  class ~ .,
  data = diabetesTrain_Scaled,
  kernel = "polynomial",
  ranges = list(cost = c(0.01, 0.05, 1, 10, 50, 100, 150, 200, 250, 300)))

plot(tune_poly, main = "")
bestmod_poly <- tune_poly$best.model

# Use the polynomial svm to predict results for the test data
poly_pred <- predict(bestmod_poly, diabetesTest_Scaled)

# Compute accuracy, sensitivity, and specificity
poly_cmat <- table(poly_pred, diabetesTest_Scaled$class)
poly_accuracy <- (poly_cmat[1,1] + poly_cmat[2,2])/sum(poly_cmat)
poly_sensitivity <- poly_cmat[2,2]/sum(poly_cmat[2,])
poly_specificity <- poly_cmat[1,1]/sum(poly_cmat[1,])
poly_misclass <- 1-poly_accuracy

# Use a radial kernel and perform cross validation to find the best cost
set.seed(2023)
tune_radial <- tune(
  svm,
  class ~ .,
  data = diabetesTrain_Scaled,
  kernel = "radial",
  ranges = list(cost = c(0.01, 0.05, 1, 10, 50, 100, 150)))

plot(tune_radial, main = "")
bestmod_radial <- tune_radial$best.model

# Use the radial svm to predict results for the test data
radial_pred <- predict(bestmod_radial, diabetesTest_Scaled)

# Compute accuracy, sensitivity, and specificity
radial_cmat <- table(radial_pred, diabetesTest_Scaled$class)
radial_accuracy <- (radial_cmat[1,1] + radial_cmat[2,2])/sum(radial_cmat)
radial_sensitivity <- radial_cmat[2,2]/sum(radial_cmat[2,])
radial_specificity <- radial_cmat[1,1]/sum(radial_cmat[1,])
radial_misclass <- 1-radial_accuracy

# Summary table of stats
model <- c("Linear", "Polynomial", "Radial")
accuracy <- c(linear_accuracy, poly_accuracy, radial_accuracy)
sensitivity <- c(linear_sensitivity, poly_sensitivity, radial_sensitivity)
specificity <- c(linear_specificity, poly_specificity, radial_specificity)
misclass <- c(linear_misclass, poly_misclass, radial_misclass)
svm_summary <- data.frame("Model" = model,
                          "Accuracy" = as.3percent(accuracy),
                          "Sensitivity" = as.3percent(sensitivity),
                          "Specificity" = as.3percent(specificity),
                          "Misclassification Rate" = as.3percent(misclass),
                          check.names = FALSE)
kable(svm_summary)
```

```r
final_summary <- rbind(tree_summary, svm_summary)
model <- c("Random forest", "SVM with RBF kernel")
accuracy <- c(dForest_accuracy, radial_accuracy)
sensitivity <- c(dForest_sensitivity, radial_sensitivity)
specificity <- c(dForest_specificity, radial_specificity)
misclass <- c(dForest_misclass, radial_misclass)
final_summary <- data.frame("Model" = model,
                            "Accuracy" = as.3percent(accuracy),
                            "Sensitivity" = as.3percent(sensitivity),
                            "Specificity" = as.3percent(specificity),
                            "Misclassification Rate" = as.3percent(misclass),
                            check.names = FALSE)
kable(final_summary)
```

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. (2013). *An introduction to statistical learning* (Vol. 112). 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. (2023). *Diabetes.* https://www.who.int/health-topics/diabetes#tab=tab_1