Predicting the risk of diabetes

STATS/CSE 780 Course Project

Pao Zhu Vivian Hsu (Student Number: 400547994)

2023-12-12

Abstract

Diabetes is a chronic disease that impacts millions of people worldwide. Machine learning techniques are being currently being used in literature to develop a risk prediction model, however more research is required to explore different techniques and improve prediction accuracy. In this paper, we use decision tree and support vector machine methods to predict the risk of diabetes. The random forest and SVM with a radial basis function kernel had the highest performance with an accuracy of 98.85% and 97.69% respectively. We found that polyuria, polydipsia, gender, and age were the most important variables related to diabetes risk. Being aware of these four variables may help healthcare professionals identify patients with a greater risk for the disease and facilitate early diagnosis.

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 is crucial 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 risk prediction model (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 size of 16 terminal nodes. To improve the accuracy and reduce the cost complexity of the tree, we used cross-validation to select the node size that produced the lowest misclassification error rate and pruned the tree accordingly. The cross-validation is illustrated in Figure 1 below. After pruning the tree, we then used random forest ensembling to further improve model performance. 1000 trees were grown and 6 predictors were sampled for splitting at each node. We chose an arbitrary large value for the number of trees and used cross-validation to select the number of predictors to split at each node. Figure 2 shows the random forest cross-validation where the number of predictors with the lowest error rate was selected.

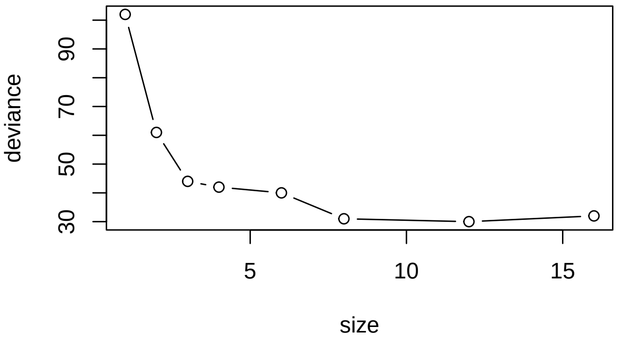

Figure 1: Cross-validation of decision tree prior to pruning; Optimal size = 12

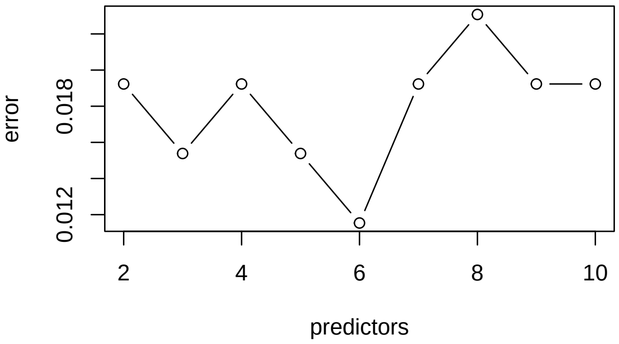

Figure 2: Cross-validation of random forest; Optimal number of predictors = 6

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, SVM is 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 3, Figure 4, and Figure 5, 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 (James et al., 2013). Moreover, a few of these metrics were used by Islam et al. (2020), so computing them allows us to directly compare our results with theirs. Comparing these four measurements between each of the methods helps us determine which model is a stronger fit to predict diabetes.

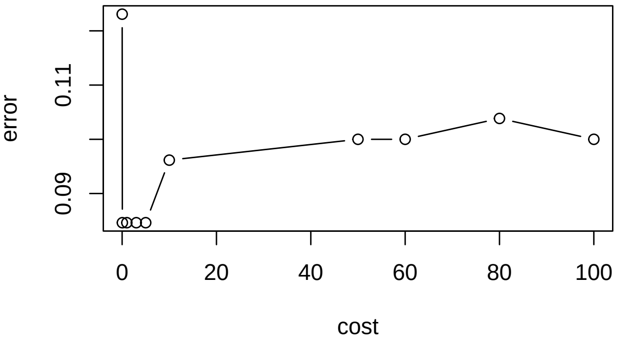

Figure 3: Cross-validation for linear SVM; Optimal cost = 3

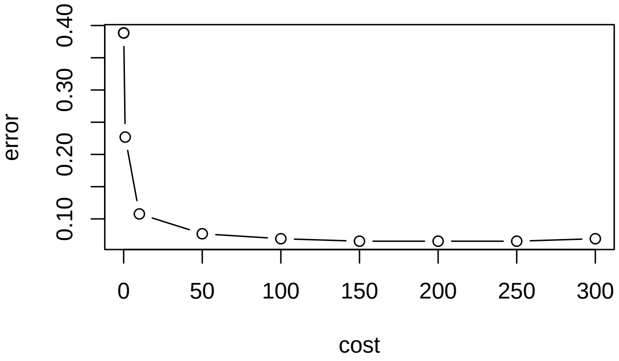

Figure 4: Cross-validation for polynomial SVM; Optimal cost = 150

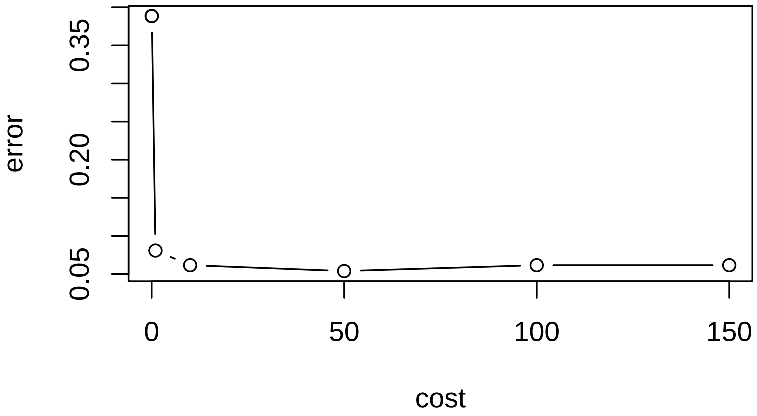

Figure 5: Cross-validation for RBF SVM; Optimal cost = 50

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

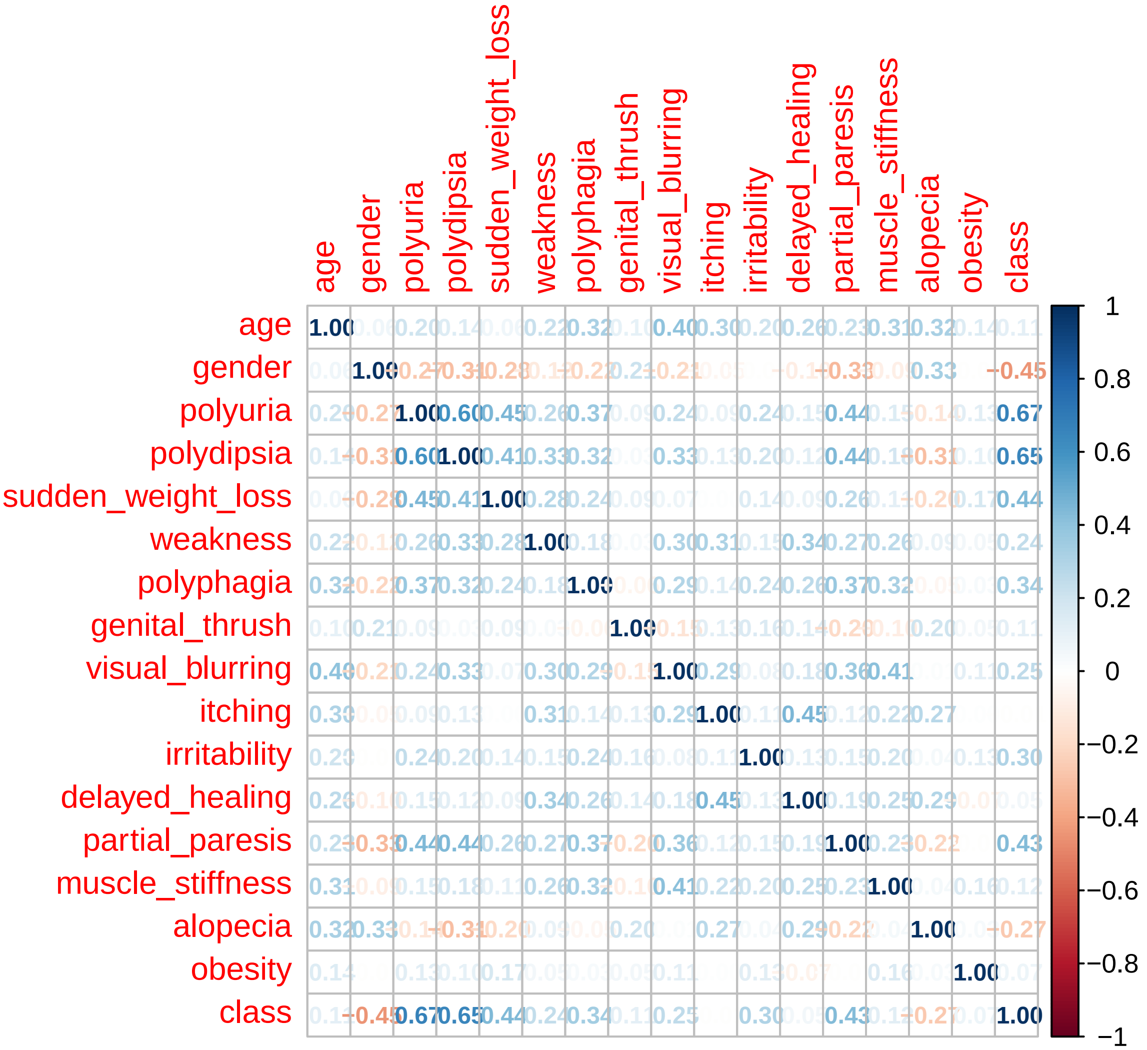

Figure 6: Correlation plot

Next, each of the individual attributes were explored. Figure 7 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.

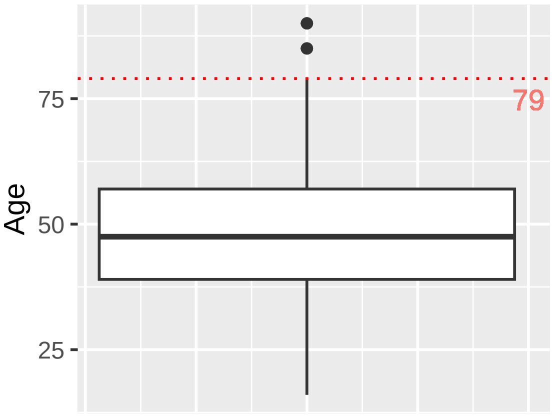

Figure 7: Boxplot of age

We also plotted the 16 categorical variables in bar charts as illustrated in Supp. Figure 10. 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 model performance. As such, we have first modelled the data in its original distribution and allowed for adjustments as required.

Decision Tree

Table 1 summarizes the results of the tree-based models. The initial tree had an accuracy of 94.62%. In the clinical setting, this means about 5 out of 100 patients would be incorrectly classified in terms of diabetes risk, which is quite high. The model's ability to correctly predict a positive case of diabetes risk is about 2.72% higher than its ability to correctly predict a negative one. An illustration of this tree is available in Supp. Figure 11.

Table 1: Model performance parameters for tree-based methods

Model	Accuracy	Sensitivity	Specificity	Misclassification Rate
Initial tree, 16 nodes	94.62%	95.65%	92.93%	5.38%
Pruned tree, 12 nodes	94.62%	95.65%	92.93%	5.38%
Random forest	98.85%	99.38%	97.98%	1.15%

While pruning the tree reduced its cost complexity, no improvements to the accuracy were observed. Nonetheless, the pruned tree gives us insight into what variables and values are most likely linked to a greater risk for diabetes as shown in Figure 8. In particular, we can see that polydipsia and gender are the first two variables to appear on the tree and are likely strong predictors for the disease. Additionally, the first internal node of the tree shows that the presence of polydipsia is likely linked with diabetes risk while the absence of the condition requires additional variables to determine risk.

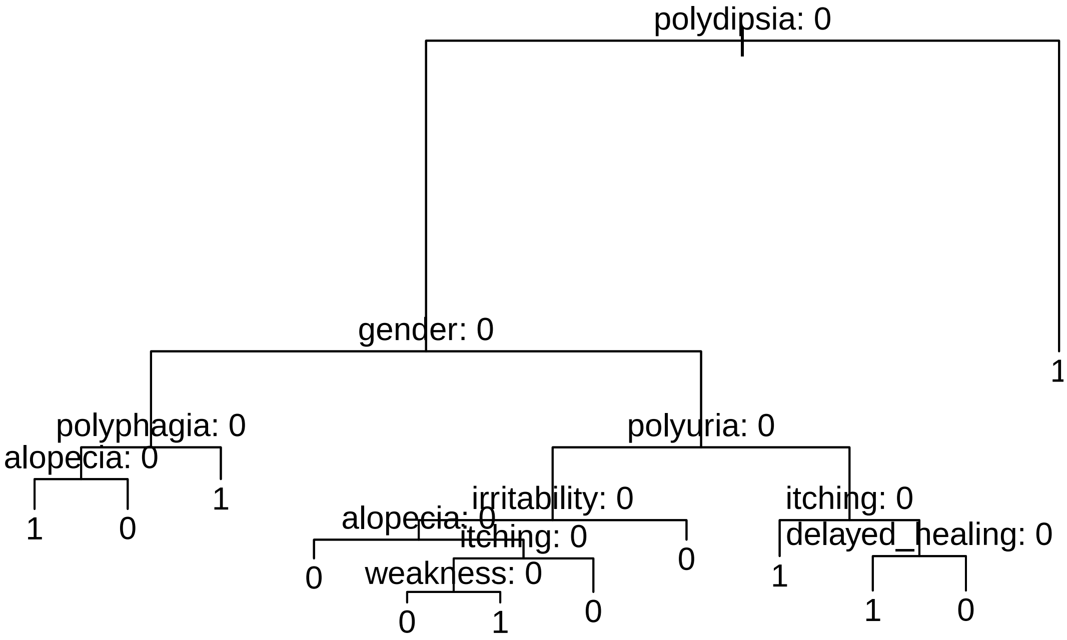

Figure 8: Pruned decision tree with 12 terminal nodes

The random forest had the best performance among the three models with an accuracy of 98.85%. This means about 1 out of 100 patients would be incorrectly classified, which is a large improvement compared to the initial tree. Its ability to predict cases of both positive and negative diabetes risk is nearly even. Additionally, we were also able to obtain a more rigorous analysis on variable importance through the random forest. We found that the most important variables in predicting diabetes include polyuria, polydipsia, gender, and age as shown in Figure 9. This is determined using both the mean decrease in accuracy and Gini index. Checking for these four variables may help healthcare professionals identify patients with a greater risk for the disease and facilitate early diagnosis.

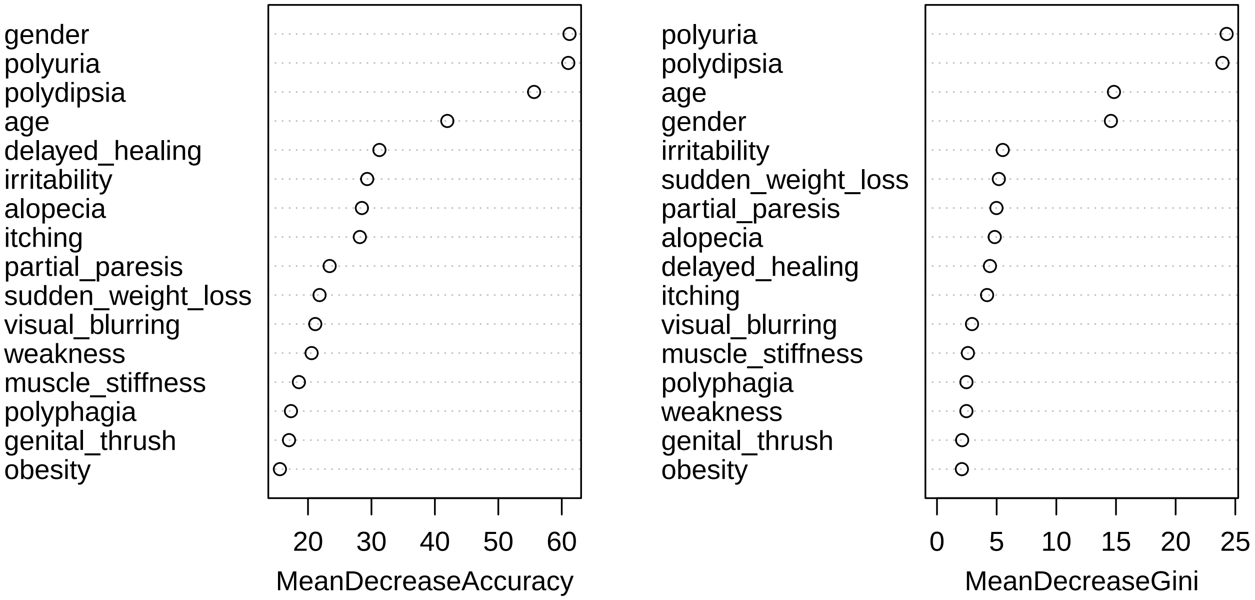

Figure 9: Importance of variables from random forest; Top is the most important and bottom is the least important

It is important to note that there is a moderate correlation between polyuria and polydipsia as observed in Figure 6 which we've presented earlier. To clarify, polyuria is a medical condition characterized by excessive urination while polydipsia is characterized by excessive water consumption (Ahmadi & Goldman, 2020; Ramirez-Guerrero et al., 2021). Since water consumption leads to urination, this could explain why both variables follow closely with one another in terms of importance. While we consider this correlation to be a moderate one, future studies could remove one of the variables or increase the number of trees in the random forest if they believe there is a large variance resulting from this correlation.

Support Vector Machine

Table 2 summarizes the performance of the SVM models. The SVM model with a linear kernel had an accuracy of 93.85%. In the clinical setting, this means about 6 out of 100 patients would be incorrectly classified in terms of diabetes risk, which is considerably high. The model's ability to predict cases of positive diabetes risk is approximately 3.11% higher than its ability to predict a negative one.

Using a polynomial kernel improved the accuracy to 95.77% and reduced the misclassification error rate to about 4 out of every 100 patients. While the model's ability to correctly predict a negative case of risk is 100%, its ability to predict a positive one is only 93.17%. This indicates that the model prediction tends to miss some cases of disease.

Finally, the SVM model with an RBF kernel had an accuracy of 97.69%, which is the highest value among the three kernels. From a clinical perspective, this means about 2 out of 100 patients would be incorrectly classified in terms of diabetes risk, which is a decent improvement compared to the linear kernel model. Compared to the polynomial kernel model, its ability to predict negative cases of risk is lower by 1.01% while its ability to predict positive cases is higher by 3.73%.

Table 2: Model performance parameters for SVM methods

Model	Accuracy	Sensitivity	Specificity	Misclassification Rate
Linear	93.85%	95.03%	91.92%	6.15%
Polynomial	95.77%	93.17%	100%	4.23%
RBF	97.69%	96.89%	98.99%	2.31%

Conclusion

Overall, each of the models in our study achieved an accuracy in the 90% to 99% range. While this may seem high, minimizing misclassification error is paramount from a medical perspective since patient health is highly dependent on accurate disease assessment. For this reason, many of the models we developed may not provide enough accuracy to be used for diabetes assessment in clinical practice. However, combining the predictions and important variables from these models along with other tools may be useful.

Table 3 summarizes the top two models developed in our study. From the table, we can see that the random forest outperforms SVM for all parameters except specificity. As a result, we can conclude that the random forest model has the highest performance compared to the remaining models. This aligns with Islam et. al's study (2020) since they have also found that their random forest methods produced the highest performance. They were able to achieve an accuracy of 97% using ten-fold cross-validation and 99% using 80:20 percentage split for their random forest models (Islam et al., 2020). These results are very similar to our random forest accuracy of 98.85%. Thus, we can say that our model results are consistent with current literature and that decision trees, specifically random forests, remain as the most accurate method for diabetes risk prediction.

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

Model	Accuracy	Sensitivity	Specificity	Misclassification Rate
Random forest	98.85%	99.38%	97.98%	1.15%
SVM with RBF kernel	97.69%	96.89%	98.99%	2.31%

For the remainder of this section, we will discuss some of the analytical challenges in this study and recommendations to improve future models. We'll begin with the random forest method where the biggest challenge we faced was ensuring stability in the results. As stated earlier, we selected an arbitrarily large value for the number of trees. This arbitrary number could be run quickly and repeatedly within the limitations of our machinery. Thus, increasing the number of trees may help improve the stability of future random forest models (James et al., 2013).

For the SVM method, selecting the best cost and kernel was our biggest challenge. The optimal choice of cost requires a balance between bias and variance (James et al., 2013). As mentioned earlier, the costs obtained through cross-validation were 3, 150, and 50 for the linear, polynomial, and RBF kernels respectively. The cost for the linear kernel model was quite low and as a result this could indicate high bias and low variance. On the other hand, the cost for the polynomial kernel model was quite high indicating that there may be a low bias and high variance. This suggests that both these models may not be a great fit with the costs selected. One way to improve this issue for future models is to apply repeated cross-validation to pick the optimal cost. This may improve the linear and polynomial kernel models and give us a better comparison between the models. Another challenge with the SVM method is selecting the best kernel. We investigated three types of kernels and selected the model with the greatest accuracy. However, this poses some risk for overfitting. Thus, further analysis on the pattern of the data could improve kernel tuning for future SVM models (James et al., 2013).

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

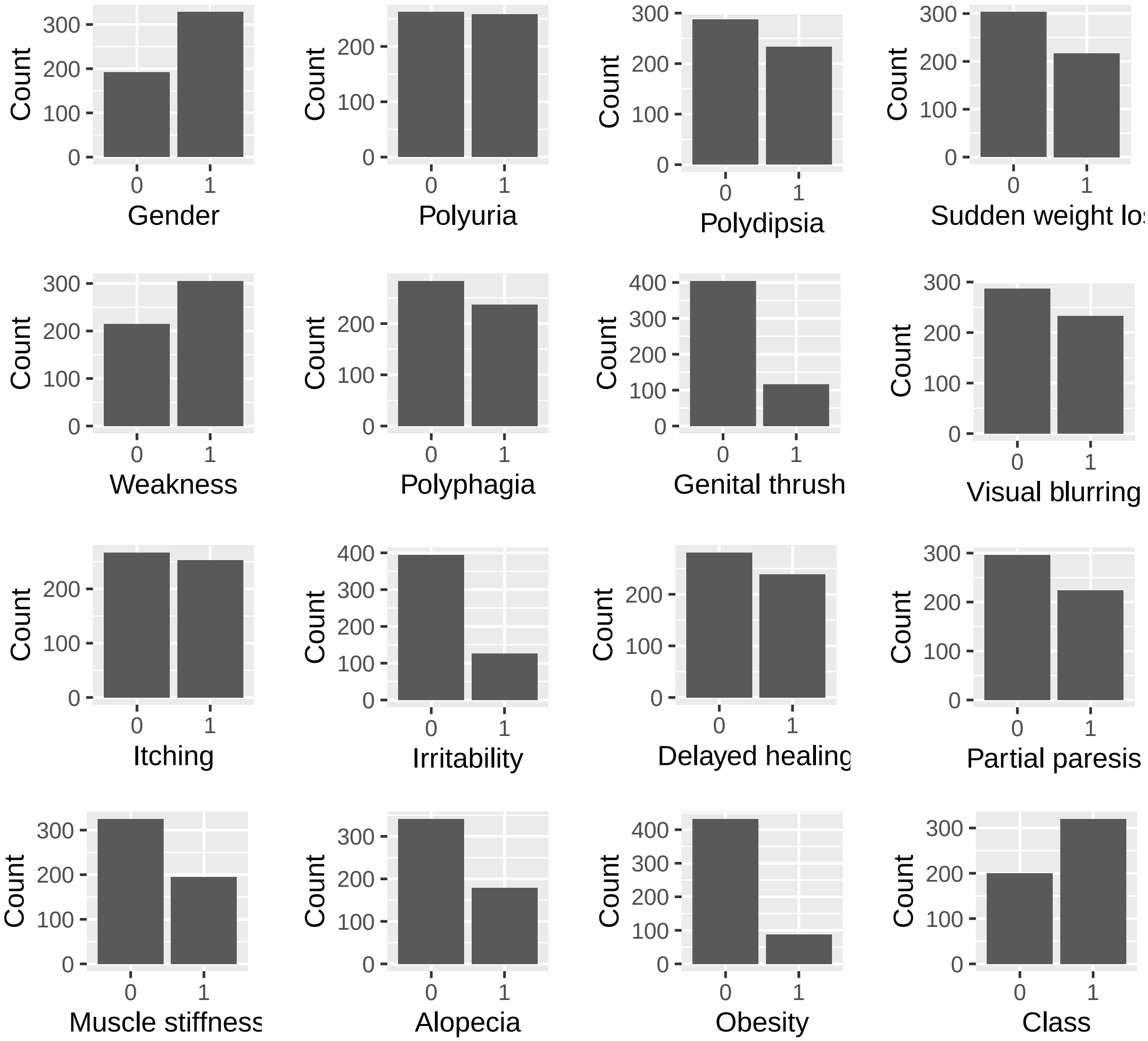

Figure 10: Bar charts of categorical variables

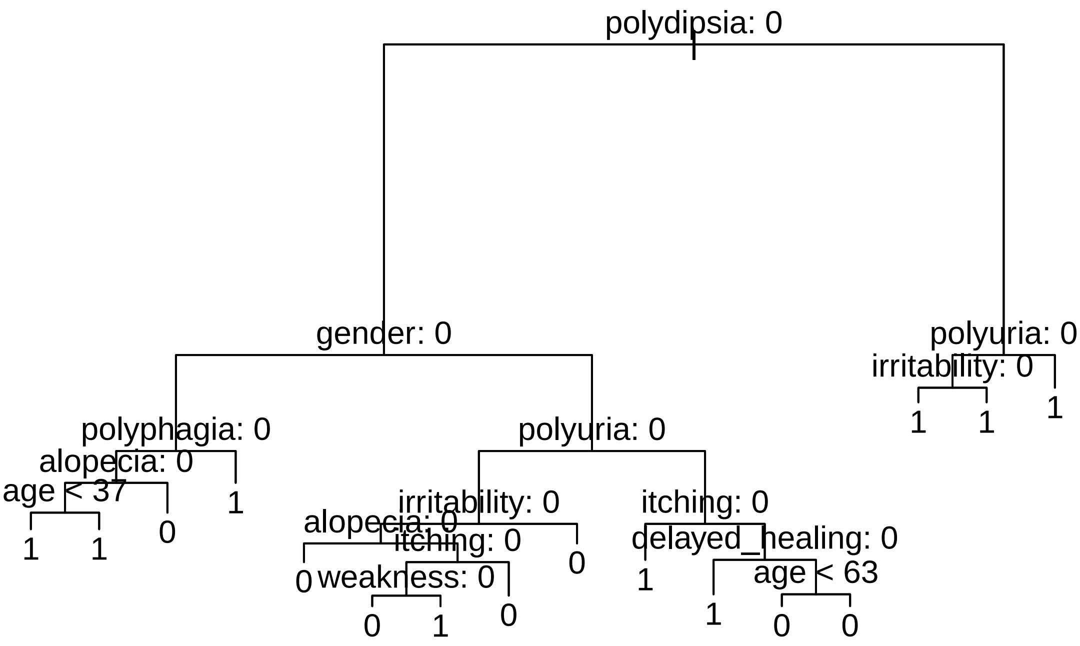

Figure 11: Initial decision tree with 16 terminal nodes

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")

# 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]

# ----- HELPER FUNCTIONS ----- #
mod.accuracy <- function(cmat) {(cmat[1,1] + cmat[2,2])/sum(cmat)}
mod.sensitivity <- function(cmat) {cmat[2,2]/sum(cmat[,2])}
mod.specificity <- function(cmat) {cmat[1,1]/sum(cmat[,1])}
as.percent <- function(value){paste0(round(value*100,2),"%")}

# ----- 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 model performance parameters
dTree_cmat <- table(pred=dTree_pred, truth=diabetesTest_class)
dTree_accuracy <- mod.accuracy(dTree_cmat)
dTree_sensitivity <- mod.sensitivity(dTree_cmat)
dTree_specificity <- mod.specificity(dTree_cmat)
dTree_misclass <- 1-dTree_accuracy
```

```r
# 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 model performance parameters
dPruneTree_cmat <- table(pred=dPruneTree_pred, truth=diabetesTest_class)
dPruneTree_accuracy <- mod.accuracy(dPruneTree_cmat)
dPruneTree_sensitivity <- mod.sensitivity(dPruneTree_cmat)
dPruneTree_specificity <- mod.specificity(dPruneTree_cmat)
dPruneTree_misclass <- 1-dPruneTree_accuracy

# Cross-validation to pick the best mtry for random forest
set.seed(2023)
cv_mtry <- c(2:10)
cv_misclass <- c()
```

```
cv_forest <- c()
for (m in cv_mtry) {
  cvForest <- randomForest(
    class ~ ., data = diabetes,
    subset = trainIndex,
    ntree = 1000,
    mtry = m)
  cvForest_pred <- predict(cvForest, newdata = diabetesTest)
  cvForest_cmat <- table(cvForest_pred, diabetesTest_class)
  cv_misclass <- c(cv_misclass, 1-mod.accuracy(cvForest_cmat))
  cv_forest <- c(cv_forest, cvForest)
}
cv_mtry_best <- cv_mtry[cv_misclass==min(cv_misclass)][1]

# Grow random forest using best mtry
dForest <- randomForest(
  class ~ ., data = diabetes,
  subset = trainIndex,
  ntree = 1000,
  mtry = cv_mtry_best,
  importance = TRUE)

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

# Compute model performance parameters
dForest_cmat <- table(pred=dForest_pred, truth=diabetesTest_class)
dForest_accuracy <- mod.accuracy(dForest_cmat)
dForest_sensitivity <- mod.sensitivity(dForest_cmat)
dForest_specificity <- mod.specificity(dForest_cmat)
dForest_misclass <- 1-dForest_accuracy
```

```r
# Plot the cross-validation on a chart
plot(cv_mtry, cv_misclass, type = "b", xlab="predictors", ylab="error")

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

# Summary table of stats
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.percent(accuracy),
                           "Sensitivity" = as.percent(sensitivity),
                           "Specificity" = as.percent(specificity),
                           "Misclassification Rate" = as.percent(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 model performance parameters
linear_cmat <- table(pred=linear_pred, truth=diabetesTest_Scaled$class)
linear_accuracy <- mod.accuracy(linear_cmat)
linear_sensitivity <- mod.sensitivity(linear_cmat)
linear_specificity <- mod.specificity(linear_cmat)
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
```

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

# Compute model performance parameters
poly_cmat <- table(pred=poly_pred, truth=diabetesTest_Scaled$class)
poly_accuracy <- mod.accuracy(poly_cmat)
poly_sensitivity <- mod.sensitivity(poly_cmat)
poly_specificity <- mod.specificity(poly_cmat)
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 model performance parameters
radial_cmat <- table(pred=radial_pred, truth=diabetesTest_Scaled$class)
radial_accuracy <- mod.accuracy(radial_cmat)
radial_sensitivity <- mod.sensitivity(radial_cmat)
radial_specificity <- mod.specificity(radial_cmat)
```

```r
radial_misclass <- 1-radial_accuracy

# Summary table of stats
model <- c("Linear", "Polynomial", "RBF")
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.percent(accuracy),
                          "Sensitivity" = as.percent(sensitivity),
                          "Specificity" = as.percent(specificity),
                          "Misclassification Rate" = as.percent(misclass),
                          check.names = FALSE)
kable(svm_summary)
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.percent(accuracy),
                            "Sensitivity" = as.percent(sensitivity),
                            "Specificity" = as.percent(specificity),
                            "Misclassification Rate" = as.percent(misclass),
                            check.names = FALSE)
kable(final_summary)
```

References

Ahmadi, L., & Goldman, M. B. (2020). *Primary polydipsia: update*. https://doi.org/10.1016/j.beem.2020.101469

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*.

Ramirez-Guerrero, G., Muller-Ortiz, H., & Pedreros-Rosales, C. (2021). *Polyuria in adults. A diagnostic approach based on pathophysiology*. https://doi.org/10.1016/j.rceng.2021.03.003

World Health Organization. (2023). *Diabetes*. https://www.who.int/health-topics/diabetes#tab=tab_1