Project 1 R program

Aryan Khanna, Baixi Jiao, and Jasmine Kellett

2024-03-21

Introduction

The goal of this analysis is to develop an effective classification model that can predict early-stage diabetes based on several medical predictor variables.

Data Loading and Preliminary Exploration

To adequately examine the dataset, we will need a few different libraries. First, we will need tidyr, dplyr, reshape2, mlr, VIM, and ggpubr. These packages are necessary to easily manipulate and analyze the data. Second, in order to adequately plot and showcase any correlations we will need corrplot and ggplot2. Lastly, we will need e1071 and caret to be able to build and test our prediction model(s).

Import Dataset

Here we will load our diabetes dataset, as well as familiarize ourselves with an overview of the data that we are working with.

```
data <- read.csv("diabetes_risk_prediction_dataset.csv")
str(data, prop=FALSE, numbers=TRUE)
```

```
## 'data.frame':    520 obs. of  17 variables:
##  $ Age               : int  40 58 41 45 60 55 57 66 67 70 ...
##  $ Gender            : chr  "Male" "Male" "Male" "Male" ...
##  $ Polyuria          : chr  "No" "No" "Yes" "No" ...
##  $ Polydipsia        : chr  "Yes" "No" "No" "No" ...
##  $ sudden.weight.loss: chr  "No" "No" "No" "Yes" ...
##  $ weakness          : chr  "Yes" "Yes" "Yes" "Yes" ...
##  $ Polyphagia        : chr  "No" "No" "Yes" "Yes" ...
##  $ Genital.thrush    : chr  "No" "No" "No" "Yes" ...
##  $ visual.blurring   : chr  "No" "Yes" "No" "No" ...
##  $ Itching           : chr  "Yes" "No" "Yes" "Yes" ...
##  $ Irritability      : chr  "No" "No" "No" "No" ...
##  $ delayed.healing   : chr  "Yes" "No" "Yes" "Yes" ...
##  $ partial.paresis   : chr  "No" "Yes" "No" "No" ...
##  $ muscle.stiffness  : chr  "Yes" "No" "Yes" "No" ...
##  $ Alopecia          : chr  "Yes" "Yes" "Yes" "No" ...
##  $ Obesity           : chr  "Yes" "No" "No" "No" ...
##  $ class             : chr  "Positive" "Positive" "Positive" "Positive" ...
```

Summary Statistics

This dataset examined diabetes through many different variables: age, gender, polyuria, polydipsia, sudden weight loss, weakness, polyphagia, gential thrush, visual blurring, itching, irritability, delayed healing, partial paresis, muscle stiffness, alopecia, obesity,and lastly, class. All of which are either symptoms or characteristics

that can point to a potential diabetes diagnosis. However, in this project we will investigate the strength of the relationship each of these characteristics has in order to build an effective model.

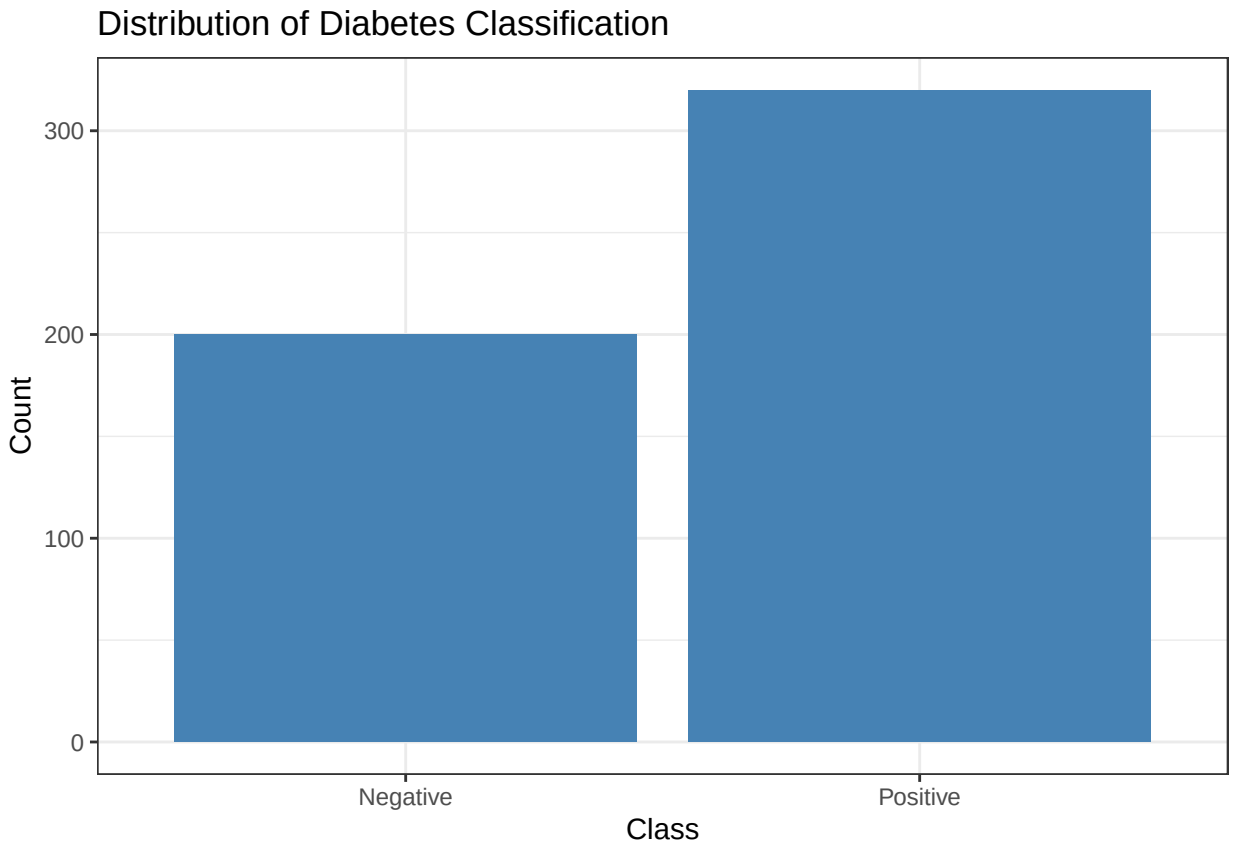
```
summary(data)
```

```
##       Age            Gender             Polyuria          Polydipsia
##  Min.   :16.00   Length:520         Length:520         Length:520
##  1st Qu.:39.00   Class :character   Class :character   Class :character
##  Median :47.50   Mode  :character   Mode  :character   Mode  :character
##  Mean   :48.03
##  3rd Qu.:57.00
##  Max.   :90.00
##  sudden.weight.loss   weakness          Polyphagia        Genital.thrush
##  Length:520         Length:520         Length:520         Length:520
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  visual.blurring      Itching          Irritability       delayed.healing
##  Length:520         Length:520         Length:520         Length:520
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  partial.paresis    muscle.stiffness     Alopecia           Obesity
##  Length:520         Length:520         Length:520         Length:520
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##     class
##  Length:520
##  Class :character
##  Mode  :character
##
##
##
```

Categorical Distribution

In the following section we will examine the distribution of participants across the variables. ### Class Distribution Here we will examine the amount of participants in this dataset that are diagnosed with diabetes versus those who are not.

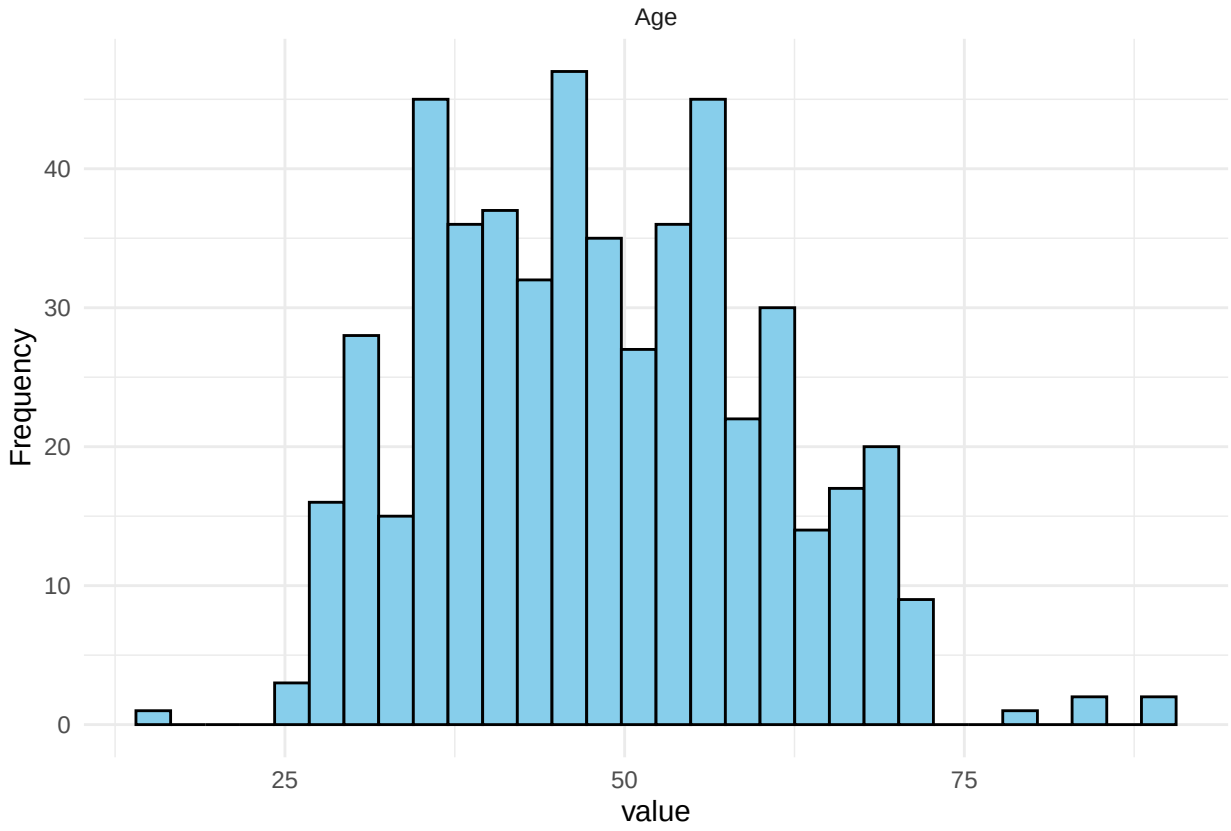
```
# Visualize the distribution of the outcome variable 'Class'
ggplot(data, aes(x = class)) +
  geom_bar(fill = 'steelblue') +
  labs(title = "Distribution of Diabetes Classification", x = "Class", y = "Count")
```



Numerical Variables Distribution

Here we will examine the ages of the participants present in the dataset. As shown in the graph below, a large majority of the participants are in the 25 - 70 age range.

```
# Assuming 'diabetes_data' is your dataset
numerical_vars <- data %>%
  select(where(is.numeric))

# Melting the data to long format for easier plotting with ggplot2
long_data <- pivot_longer(numerical_vars, cols = everything())

# Plotting
ggplot(long_data, aes(x = value)) +
  geom_histogram(bins = 30, fill = 'skyblue', color = 'black') +
  theme_minimal() +
  facet_wrap(~name, scales = 'free') +
  labs(y = "Frequency")
```



Relationship Between Attributes and Class

Here we will examine the relationship different attributes have with the criteria of being diagnosed with diabetes. Similar to the previous section, we will need to convert character variables to factors and use the pivot longer function. However, for this analysis we will be looking at the proportions of each attribute and the criteria of a diabetes diagnosis.

```
diabetes_data <- data %>%
  mutate_if(is.character, as.factor)

# Convert dataset to long format
long_data <- pivot_longer(diabetes_data, cols = -c(Age, class))

# Generate plots
ggplot(long_data, aes(x = value, fill = class)) +
  geom_bar(position = "fill") +
  theme_minimal() +
  labs(y = "Proportion") +
  facet_wrap(~name, scales = "free_x", nrow = 2) +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 45, hjust = 1),
        legend.title = element_text(size = 12),
        legend.text = element_text(size = 10))
```

Here we discover higher proportions within the positive diabetes criteria, for the attributes: gender, polydispia, polyphagia, sudden weight loss, partial paresis, and general weakness. This finding is to be expected since all of the variables listed thus far and known to have some relationship with a positive diagnosis. However, the proportion within the alopecia category was suprising. Since both type 1 and 2 diabetes often induces hair loss and makes individuals prone to developing alopecia conditions.

Missing Values

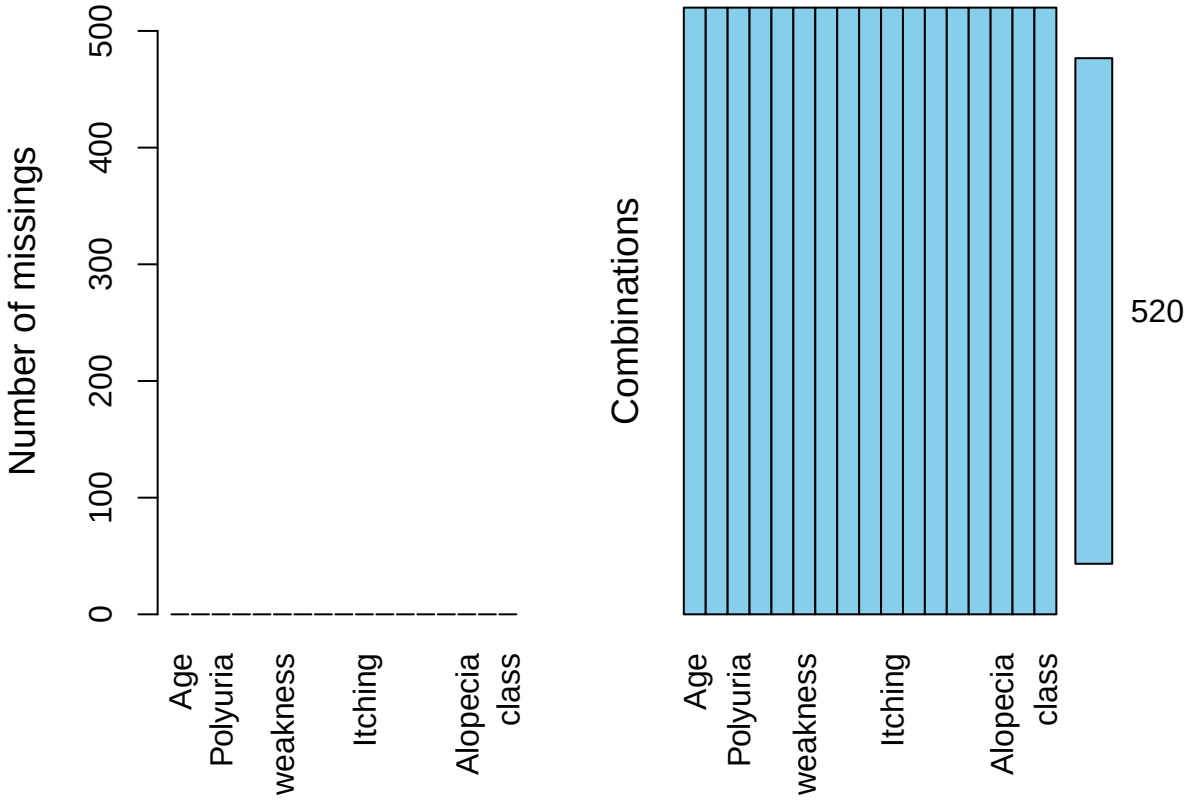
Here we will examine if we have to worry about an missing values that could be present in the data. Fortunately, no such values were present.

```
data %>%
  mutate_if(is.character,as.factor) %>%
  summary()
```

```
##       Age           Gender    Polyuria  Polydipsia sudden.weight.loss weakness
##  Min.   :16.00   Female:192   No :262   No :287    No :303             No :215
##  1st Qu.:39.00   Male  :328   Yes:258   Yes:233    Yes:217             Yes:305
##  Median :47.50
##  Mean   :48.03
##  3rd Qu.:57.00
##  Max.   :90.00
##  Polyphagia Genital.thrush visual.blurring Itching   Irritability
##  No :283    No :404        No :287         No :267   No :394
##  Yes:237    Yes:116        Yes:233         Yes:253   Yes:126
##
##
##
##
```

```
##  delayed.healing partial.paresis muscle.stiffness Alopecia  Obesity
##  No :281         No :296         No :325          No :341   No :432
##  Yes:239         Yes:224         Yes:195          Yes:179   Yes: 88
##
##
##
##
##       class
##  Negative:200
##  Positive:320
##
##
##
##
```

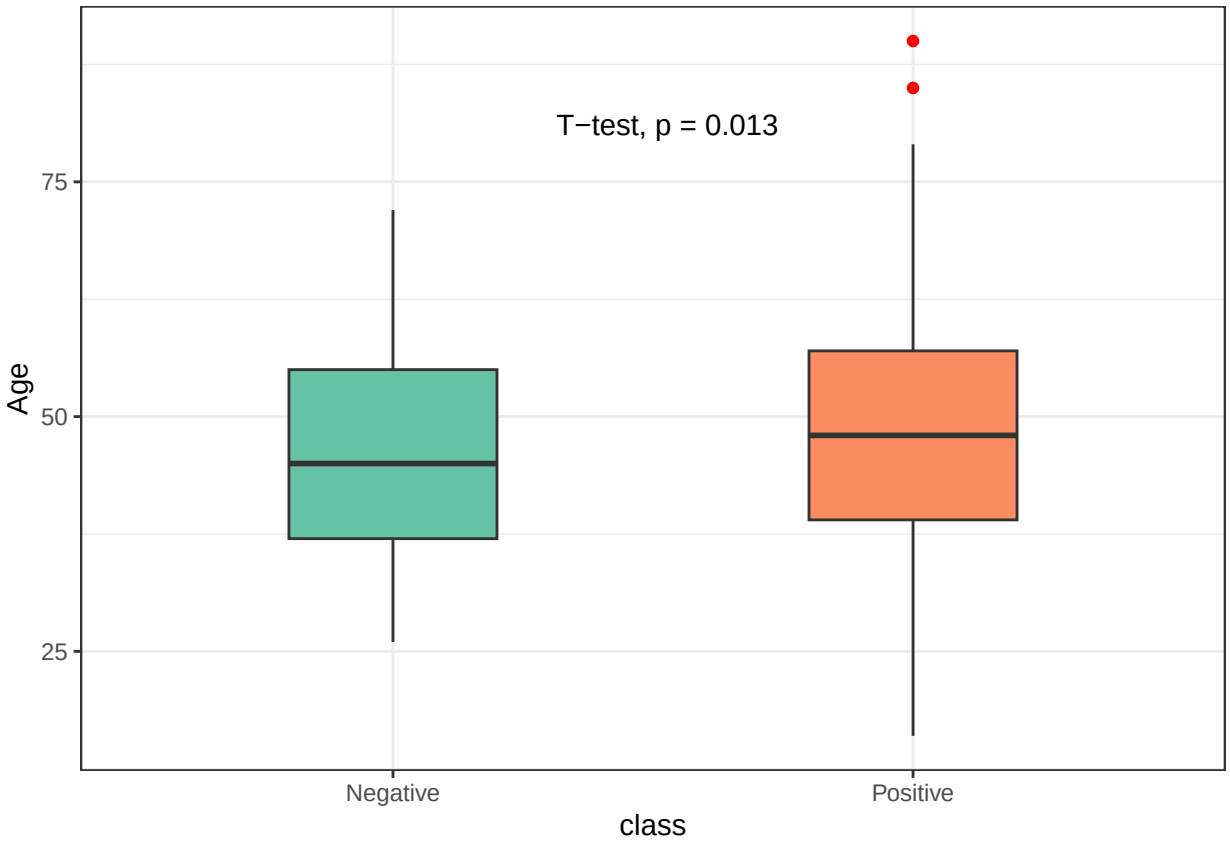
```
aggr(data, delimiter="_imp",prop=FALSE, numbers=TRUE)
```



There are a total of 520 observations and 17 features in the original data, of which only the variable `Age` is a numerical variable, and the rest are binary variables,and there are no missing values in the data.

Boxplot Examining Age and Class

```
data %>% ggplot(aes(class,Age,fill = class)) +
  geom_boxplot(width = 0.4,outlier.color = "red") +
  scale_fill_brewer(palette = "Set2" ) +
  theme(legend.position = "none") +
  stat_compare_means(method = "t.test",label.x = 1.4, label.y = 80)
```

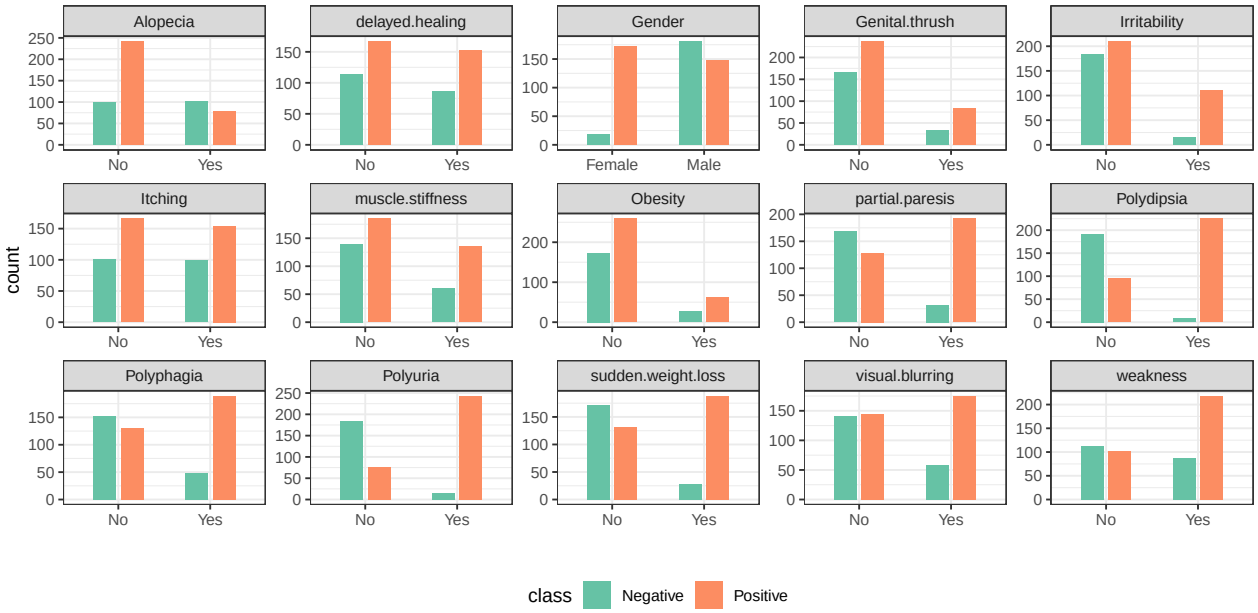


Based on the observations of the grouped boxplots and the results of the t-test, we found that age was significant for the predicted categories. Specifically, there is a difference in the medians of the negative and positive age groups, with the positive group having a larger median than the negative group. This suggests that age plays an important role in differentiating categories.

Significance Testing

In order to get to the heart of the nature of these attributes' relationships in relation to diabetes, we will create a series of bar graphs. Here we will also try to eliminate some factors are not crucial to our testing.

```
data %>%
  select(-1) %>%
  pivot_longer(cols = 1:15) %>%
  ggplot(aes(x=value,fill = class)) +
  geom_bar(position = position_dodge(width = 0.6),width = 0.5) +
  labs(x = "") +
  scale_fill_brewer(palette = "Set2") +
  facet_wrap(~name,scales = "free",ncol = 5) +
  theme(legend.position = "bottom")
```



The variables {delayed.healing}, {Genital.thrush}, {Obesity}, and {ltching} do not significantly differ in the proportion of distribution in the various categories, as indicated by the grouped bar chart. This implies that these four factors might not be very significant in terms of prediction.

```
with(data,chisq.test(delayed.healing,class))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  delayed.healing and class
## X-squared = 0.96209, df = 1, p-value = 0.3267
```

```
with(data,chisq.test(Genital.thrush,class))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  Genital.thrush and class
## X-squared = 5.7921, df = 1, p-value = 0.0161
```

```
with(data,chisq.test(Itching,class))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  Itching and class
## X-squared = 0.046235, df = 1, p-value = 0.8297
```

```
with(data,chisq.test(Obesity,class))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  Obesity and class
## X-squared = 2.3275, df = 1, p-value = 0.1271
```

With the exception of the variable {Genital.thrush}, the chi-square test findings provide additional evidence in favor of the conclusions drawn in the bar chart. Since {Genital.thrush} has a substantially different distribution

across categories (p-value less than 0.05), it cannot be ruled out as a predictor variable. Consequently, we are limited to ruling out the final three variables: 'delayed.healing', 'Obesity', and 'ltching'.

Data processing

```
data_reduce <- data %>%
  select(-c('delayed.healing','Obesity','Itching','class')) %>%
  mutate_if(is.character,as.factor) %>%
  fastDummies::dummy_cols() %>%
  select(-c(2:13)) %>%
  mutate(class = as.factor(data$class))

set.seed(123)
index <- 1:nrow(data_reduce)
test_set_index <- sample(index, trunc(length(index)/3))
test_set <- data_reduce[test_set_index,]
train_set <- data_reduce[-test_set_index,]
```

SVM

Here we will explore our SVM model, also known as support vector machine model. SVM models are supervised machine learning models which can be used to solve a multitude of problems by performing data transformations. It is our hope that this model will effectively make a prediction model.

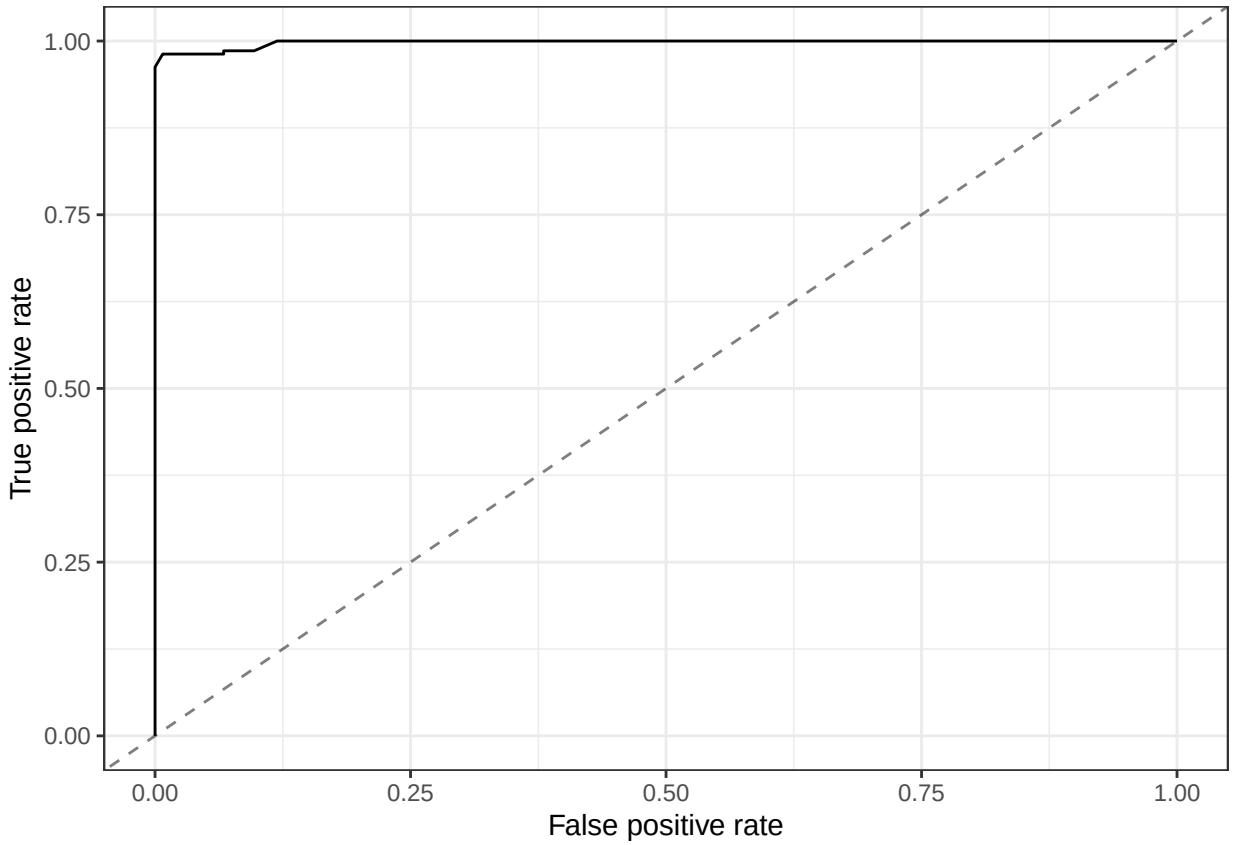
```
set.seed(123)
# create a classification task
task_f <- makeClassifTask(id = "diabetes_class_F",
                          data = train_set,
                          target = "class",
                          positive = "Positive")
# create a svm learner
svm_lrn_f <- makeLearner("classif.svm",
                         id = "svm_full",
                         predict.type = "prob")

svm_mod_f <- train(svm_lrn_f,task_f)
train_svm_f <- predict(svm_mod_f, task_f)
test_svm_f <- predict(svm_mod_f, newdata=test_set);

cat("Training set accuracy: ", performance(train_svm_f, measures=acc), "\n")
```

```
## Training set accuracy:  0.9769452
```

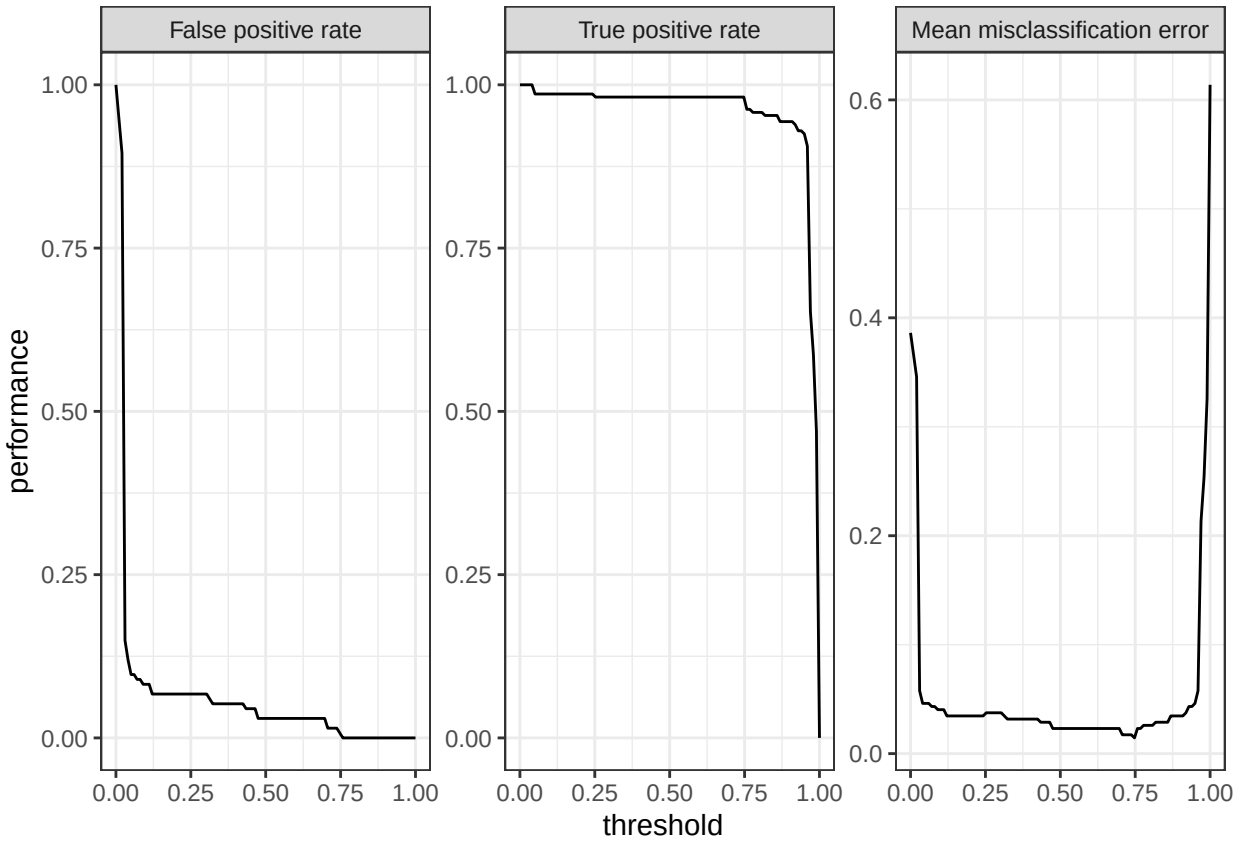
```
cat("Test set accuracy: ", performance(test_svm_f, measures=acc), "\n")
```

```
## Test set accuracy:  0.9537572
```

```
d = generateThreshVsPerfData(train_svm_f, measures = list(fpr, tpr, mmce))
plotROCCurves(d)
```




```
plotThreshVsPerf(d)
```



According to the above two figures, the training set's AUC for the support vector machine with its default parameters is essentially close to 1, and the threshold of 0.5 was chosen appropriately. At this point, the training set's accuracy is 0.9769452, while the test set's accuracy rate is 0.9537572.

Here begin resampling the data and setting discrete parameters.

```
set.seed(123)
# Define the resampling strategy
rdesc <- makeResampleDesc(method = "CV", iters = 10)
# discrete parameter sets
discrete_ps <- makeParamSet(
  makeNumericParam("cost", lower = 0.1, upper = 3),
  makeNumericParam("gamma", lower = 0.1, upper = 3)
)

ctrl_d <- makeTuneControlGrid()

res_svm <- tuneParams(svm_lrn_f,
                      task = task_f,
                      resampling = rdesc,
                      par.set = discrete_ps,
                      control = ctrl_d,
                      measures = list(acc, mmce),
                      show.info = FALSE)
res_svm
```

```
## Tune result:
## Op. pars: cost=0.422; gamma=0.1
```

```
## acc.test.mean=0.9653782,mmce.test.mean=0.0346218
```

The ideal settings for the 10-fold cross-validation are {cost=0.422; gamma=0.1}, and the associated average accuracy rate is 0.9653782.

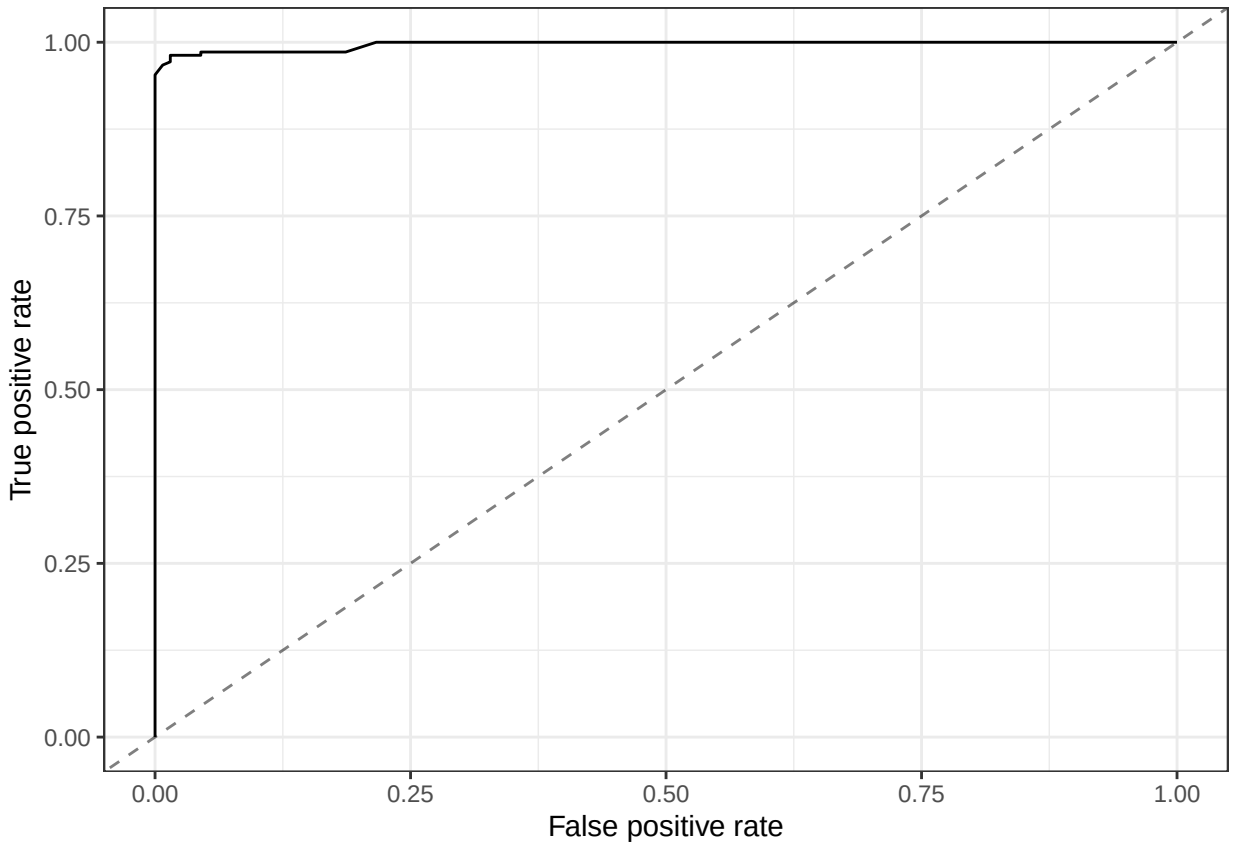
```
set.seed(123)
svm_lrn_f_tuned <- setHyperPars(svm_lrn_f, par.vals = res_svm$x)
svm_mod_f_tuned <- train(svm_lrn_f_tuned, task_f)
train_svm_f_tuned <- predict(svm_mod_f_tuned, task_f);
test_svm_f_tuned <- predict(svm_mod_f_tuned, newdata=test_set);

cat("Training set accuracy: ", performance(train_svm_f_tuned, measures=acc), "\n")
```

```
## Training set accuracy:  0.9711816
```

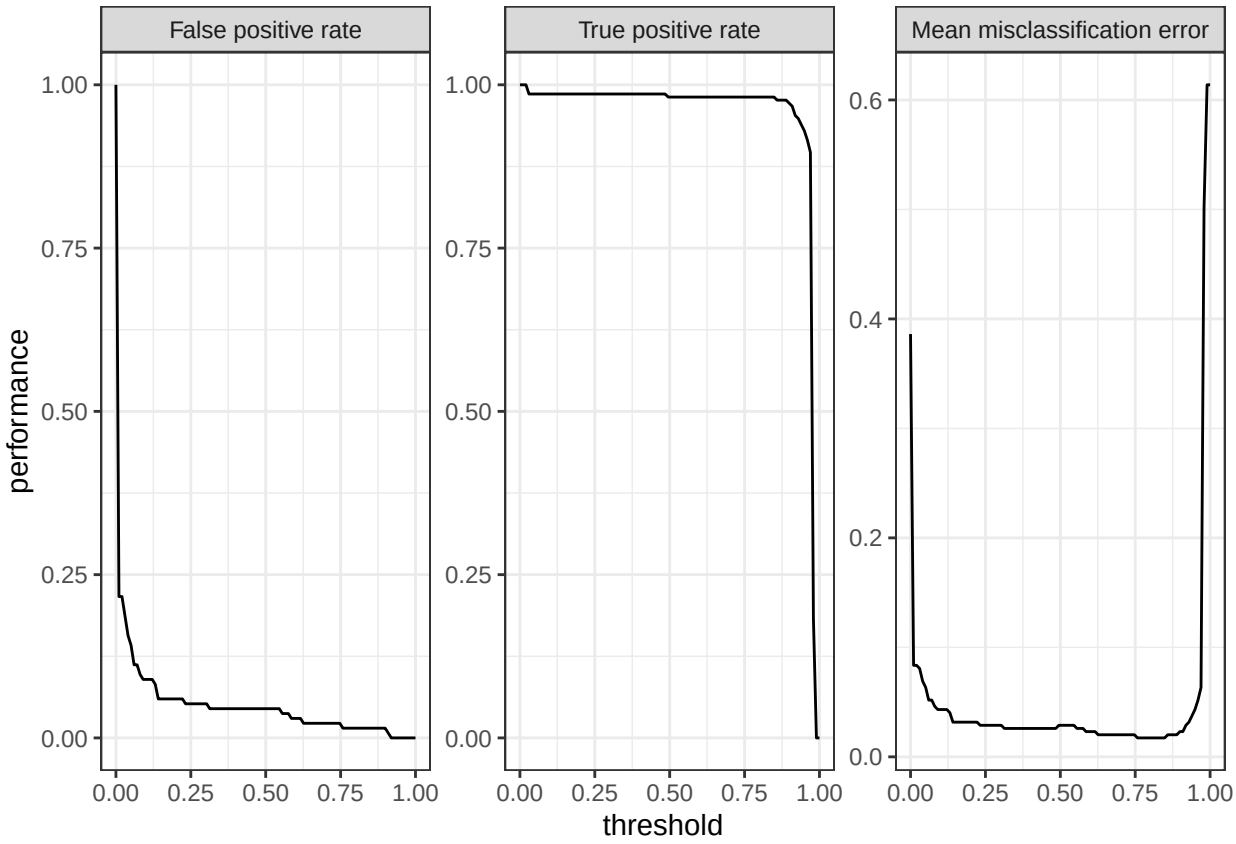
```
cat("Test set accuracy: ", performance(test_svm_f_tuned, measures=acc), "\n")
```

```
## Test set accuracy:  0.9595376
```

```
d = generateThreshVsPerfData(train_svm_f_tuned, measures = list(fpr, tpr, mmce))
plotROCCurves(d)
```




```
plotThreshVsPerf(d)
```



The accuracy rate on the test set is 0.9595376, while the accuracy rate on the training set is 0.9711816, as can be observed from the above findings; the performance of the optimized SVM is marginally better on the test set. This is because there is less overfitting on the training set, which increases accuracy and generalization capacity on the test set.

Neural Network

Here we attempt to build a second model which may be more effective than the SVM model.

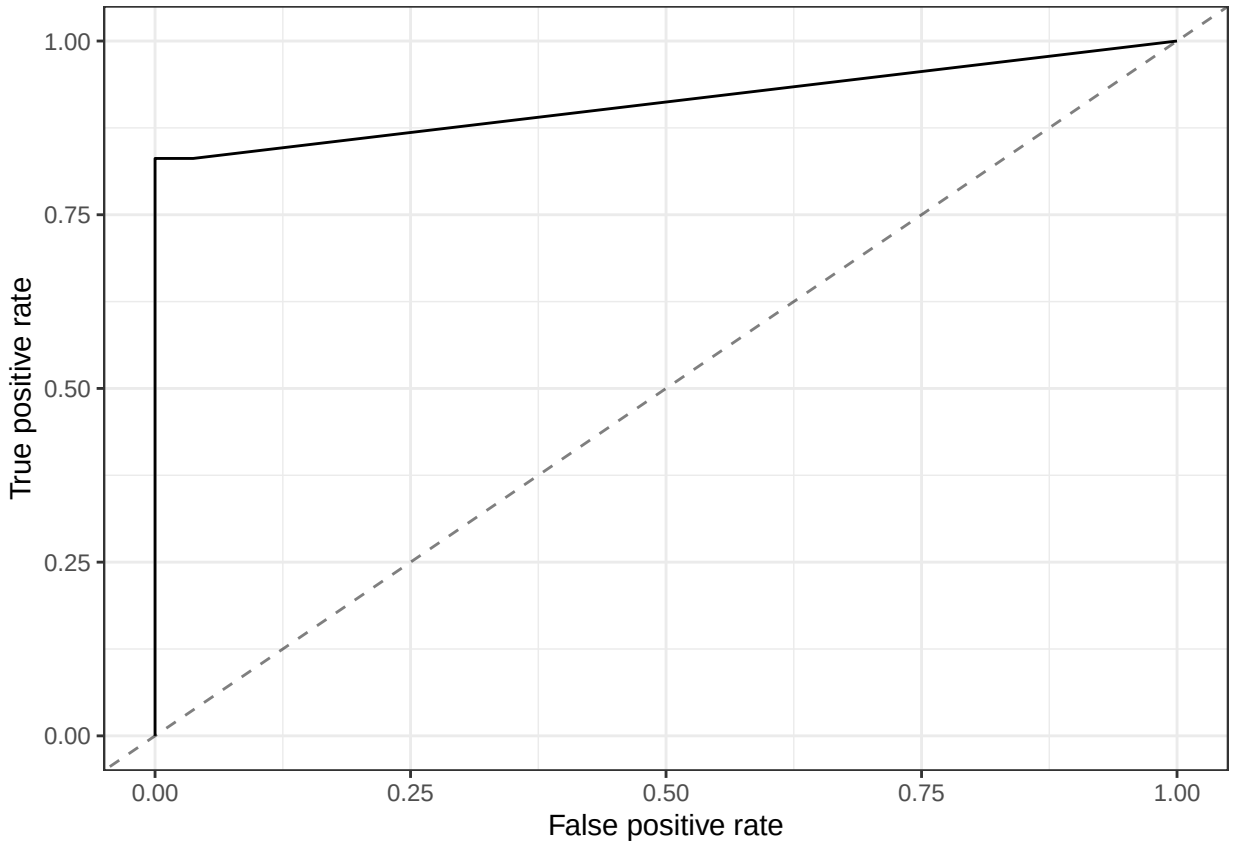
```
set.seed(123)
nn_lrn = makeLearner("classif.nnet",
                     predict.type = "prob")

nn_mod <- train(nn_lrn,task_f)
train_nn <- predict(nn_mod, task_f)
test_nn <- predict(nn_mod, newdata = test_set)

cat("Training set accuracy: ", performance(train_nn, measures = acc), "\n")
```

```
## Training set accuracy:  0.8962536
```

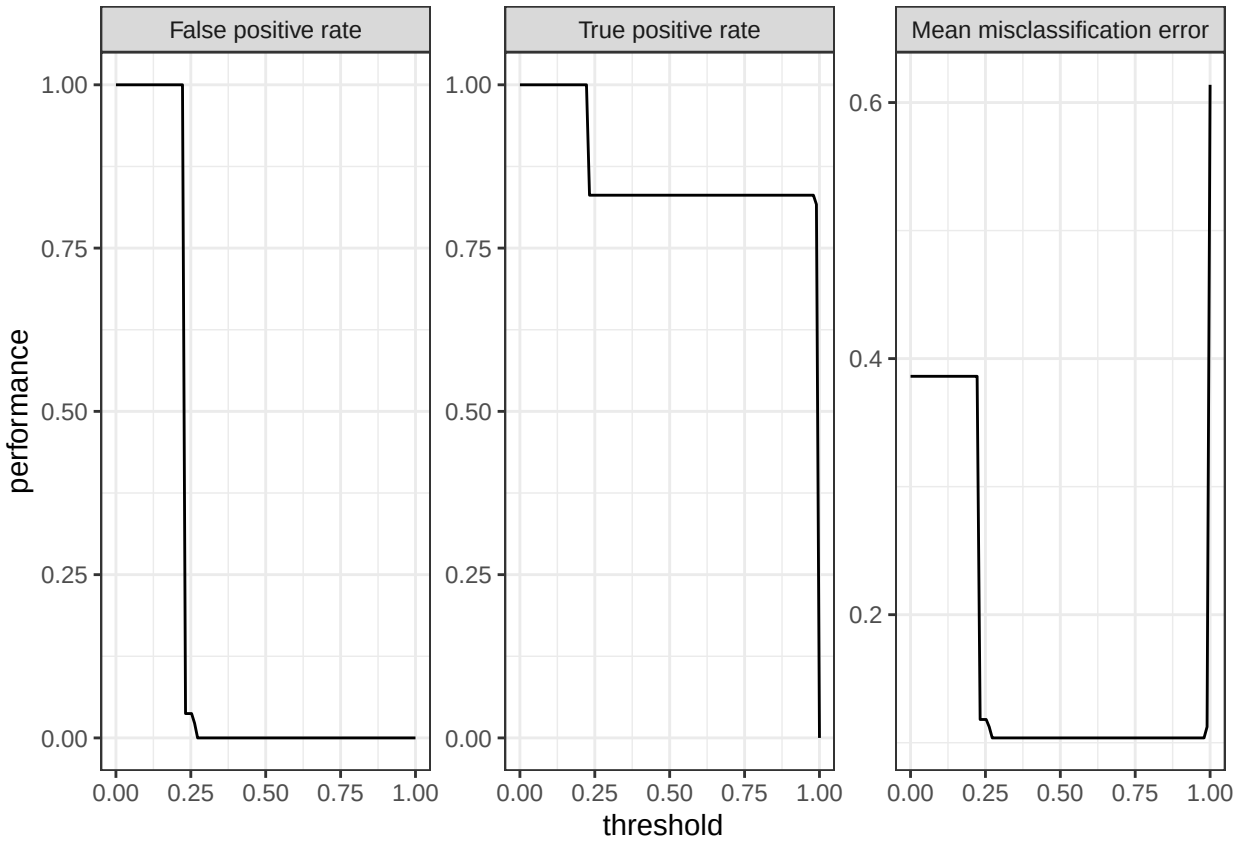
```
cat("Test set accuracy: ", performance(test_nn, measures = acc), "\n")
```

```
## Test set accuracy:  0.8843931
```

```
d = generateThreshVsPerfData(train_nn, measures = list(fpr, tpr, mmce))
plotROCCurves(d)
```




```
plotThreshVsPerf(d)
```



The aforementioned data demonstrate how much poorer the neural network's default performance is than the support vector machine's. As of right now, the test set's accuracy rate is 0.8843931, while the training set's accuracy rate is 0.8962536.

```
# getParamSet(makeLearner("classif.nnet"))

# Define the resampling strategy
set.seed(123)

# discrete parameter sets
discrete_ps <- makeParamSet(
  makeDiscreteParam("size", values = c(2:10)),
  makeDiscreteParam("decay", values = 10^-(1:5)),
  makeDiscreteParam("maxit", values = 10000L)
)

res_nn <- tuneParams(nn_lrn,
                     task = task_f,
                     resampling = rdesc,
                     par.set = discrete_ps,
                     control = ctrl_d,
                     measures = list(acc, mmce))
res_nn
```

```
## Tune result:
## Op. pars: size=10; decay=0.001; maxit=10000
## acc.test.mean=0.9684874,mmce.test.mean=0.0315126
```

For the neural network, I chose to optimize three parameters. Under 10-fold cross-validation, the optimal parameter combination is `size=10; decay=0.001; maxit=10000`, and the corresponding average accuracy rate is 0.9684874.

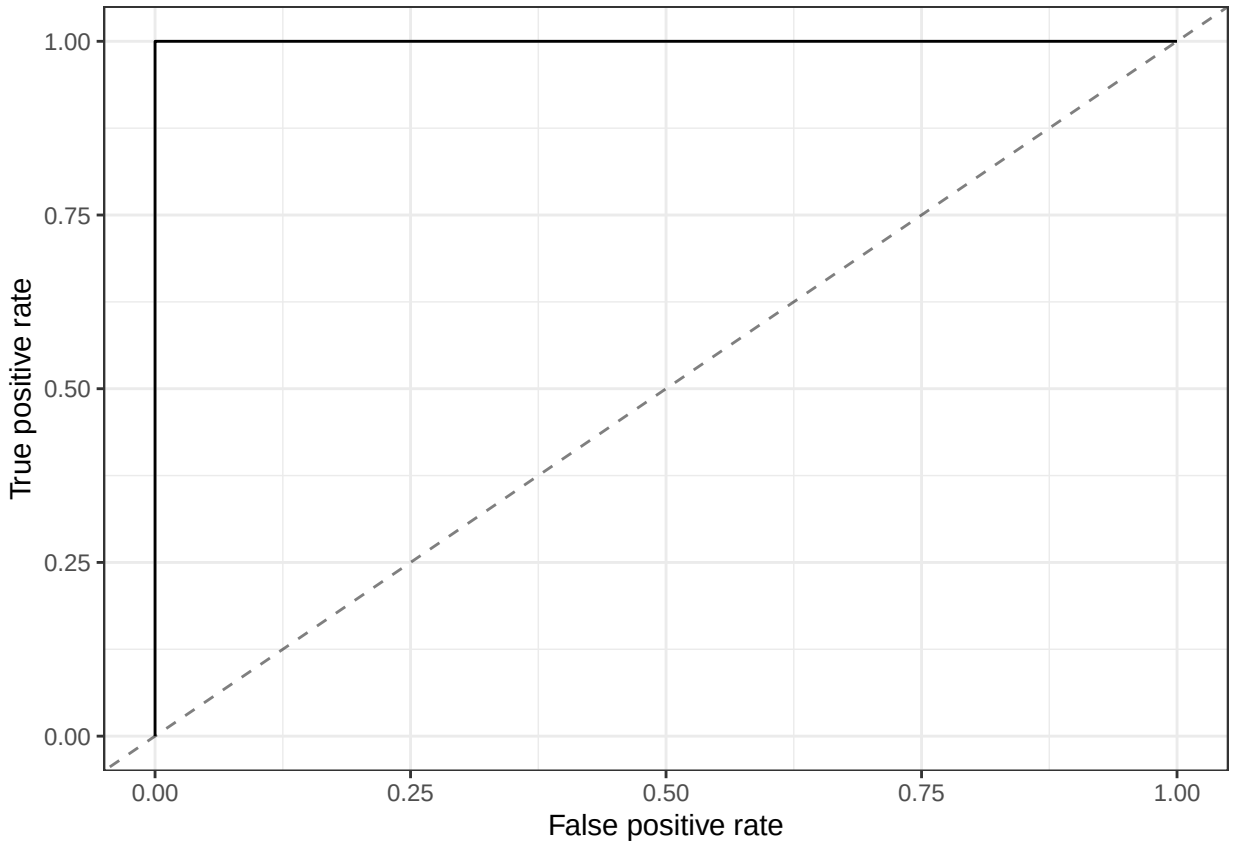
```
set.seed(123)
nn_lrn_tuned <- setHyperPars(nn_lrn, par.vals = res_nn$x)
nn_mod_tuned <- train(nn_lrn_tuned, task_f)
train_nn_tuned <- predict(nn_mod_tuned, task_f);
test_nnf_tuned <- predict(nn_mod_tuned, newdata=test_set);

cat("Training set accuracy: ", performance(train_nn_tuned, measures=acc), "\n")
```

```
## Training set accuracy:  1
```

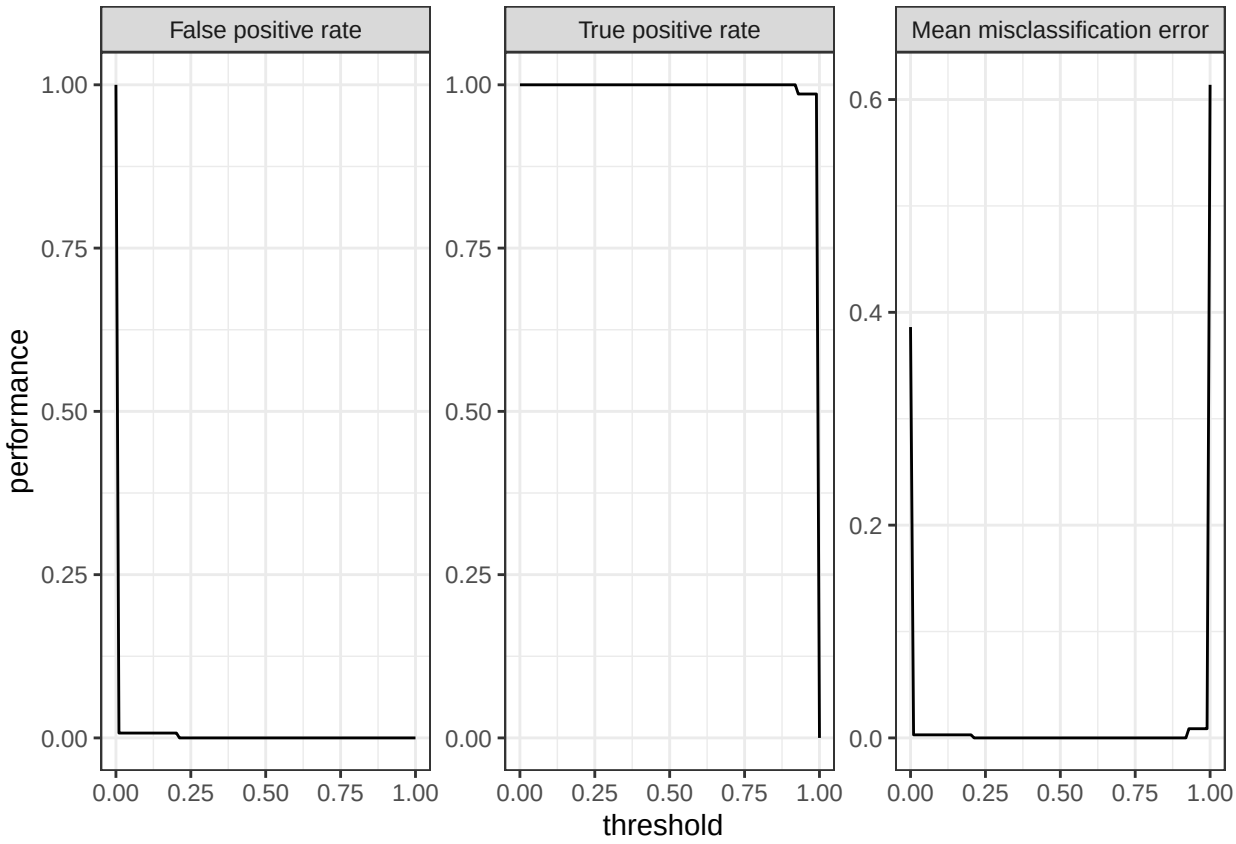
```
cat("Test set accuracy: ", performance(test_nnf_tuned, measures=acc), "\n")
```

```
## Test set accuracy:  0.9248555
```

```
d = generateThreshVsPerfData(train_nn_tuned, measures = list(fpr, tpr, mmce))
plotROCCurves(d)
```




```
plotThreshVsPerf(d)
```



The optimized neural network performs better than the neural network with default settings (size = 3), with an accuracy rate of 1 on the training set and 0.9248555 on the test set.

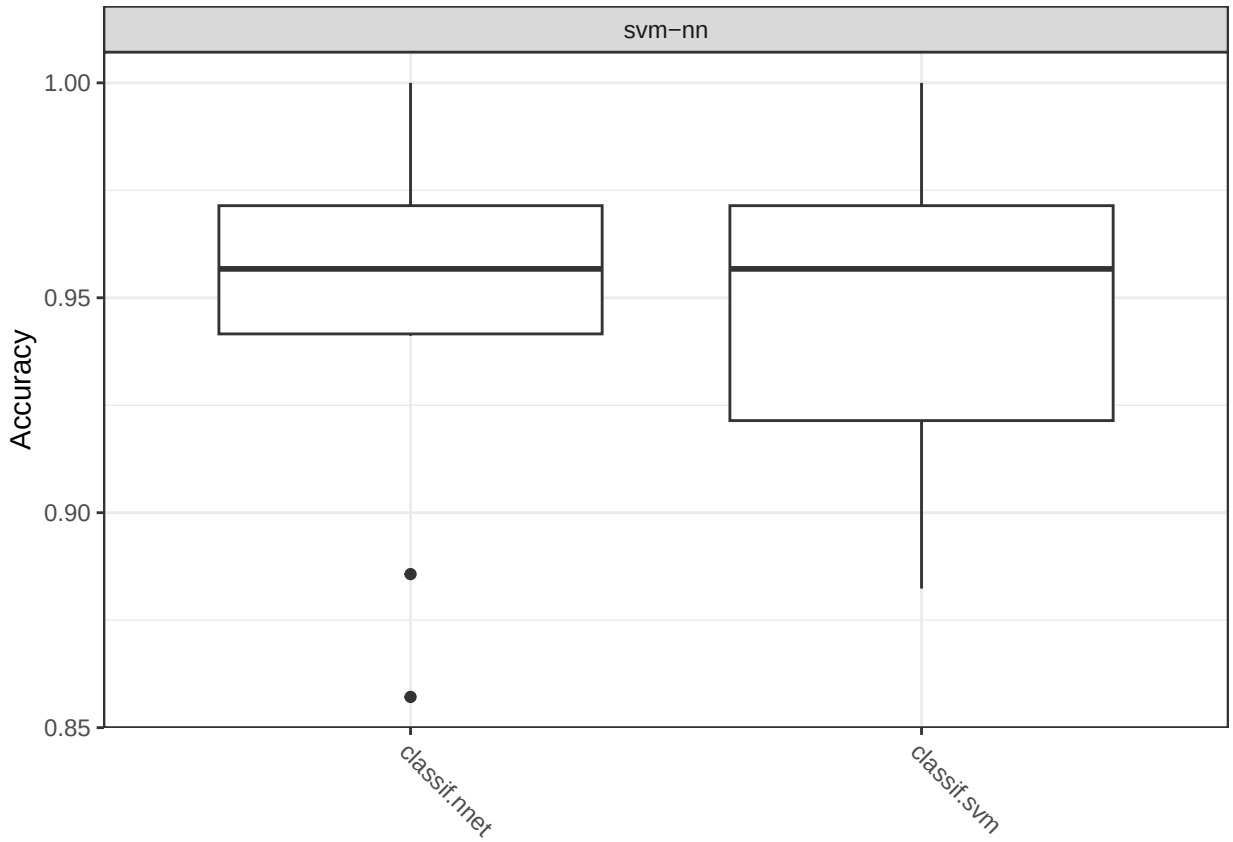
Do evaluating between 2 models

Here we will be begin our analysis on the comparison of both models to find the most effective one.

```
set.seed(123)
# create benchmark tasks
svm_nn_task <- makeClassifTask(id = "svm-nn",
                               data = train_set,
                               target = "class",
                               positive = "Positive")
# create learners for svm and nn
lrns = list(makeLearner("classif.svm", kernel = "radial", cost=0.422, gamma=0.1),
            makeLearner("classif.nnet", size=10, decay=0.001, maxit=10000L))

# conduct the benchmark
bmr = benchmark(lrns, svm_nn_task, rdesc, measures=list(acc, mmce, ber))
bmr
```

```
##   task.id   learner.id acc.test.mean mmce.test.mean ber.test.mean
## 1  svm-nn  classif.svm     0.9481513     0.05184874    0.06165036
## 2  svm-nn classif.nnet     0.9483193     0.05168067    0.05989528
```

```
plotBMRBoxplots(bmr, measure = acc)
```

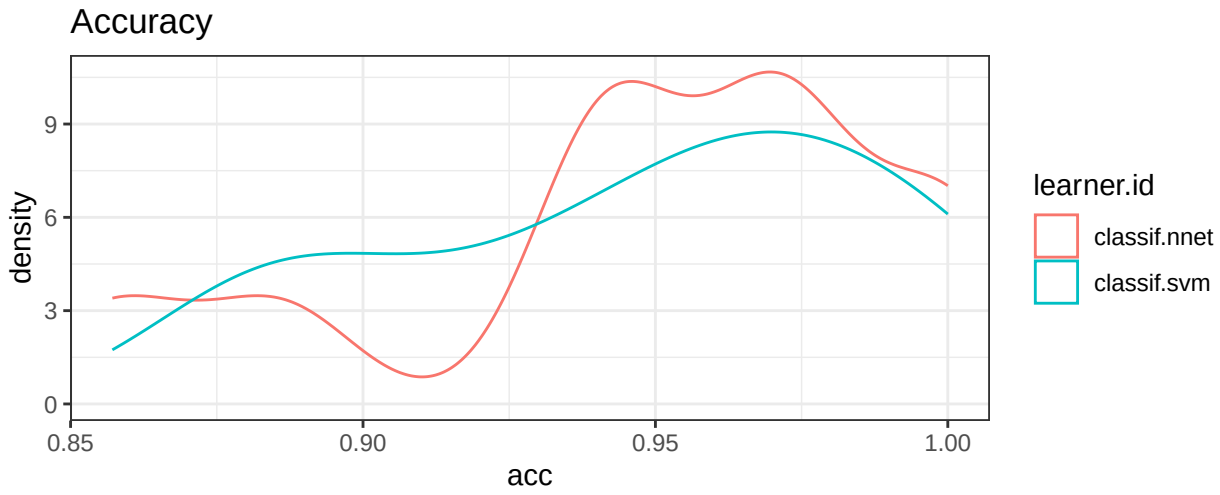


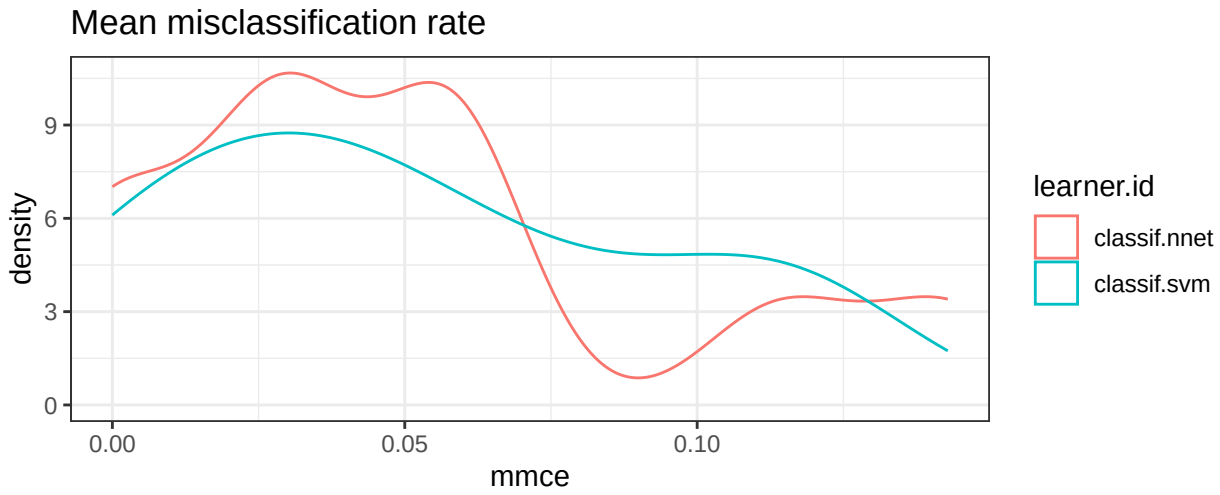

```
perf <- getBMRPerformances(bmr, as.df=TRUE)
p1<-ggplot(perf, aes(acc, colour = learner.id)) +
  geom_density() +
  labs(title="Accuracy")

p2<-ggplot(perf, aes(mmce, colour = learner.id)) +
  geom_density() +
  labs(title="Mean misclassification rate")
p3<-ggplot(perf, aes(ber, colour = learner.id)) +
  geom_density() +
  labs(title="Balanced error rate")
library(gridExtra)
```

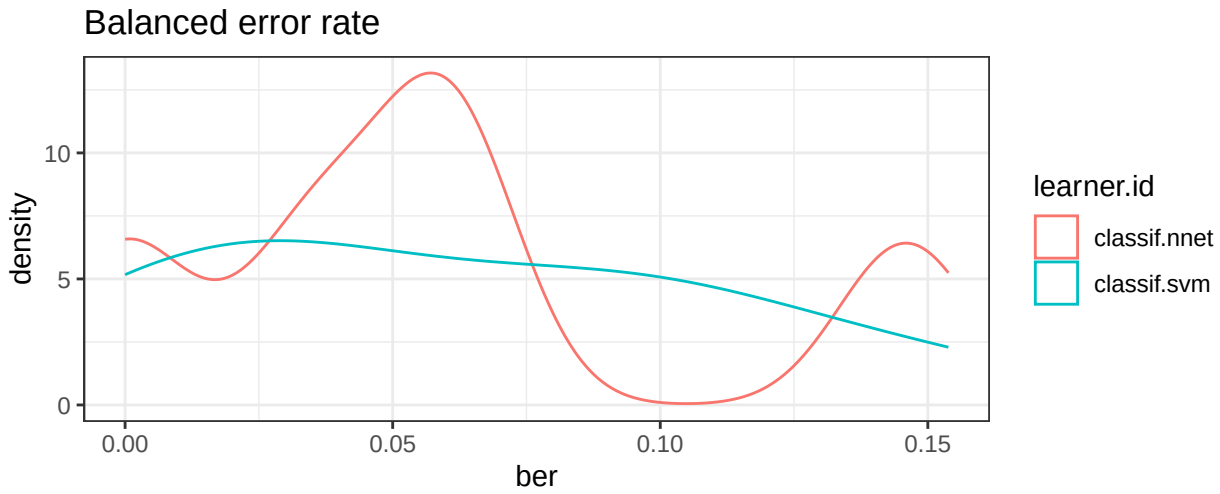
```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
grid.arrange(p1,p2,p3,ncol=1)
```







Based on the outcomes of 10-fold cross-validation, it can be inferred that the two models' performance on the training set is almost identical. The neural network performs marginally better, although it frequently performs worse on the test set because to its propensity for overfitting.

Report

We found connections between 16 factors and the health state of the patient through data investigation. Three features—"ltching," "delayed.healing," and "Obesity"—were determined to be unrelated to predictions by statistical testing and visualization. We chose two machine learning models—a neural network and a support vector machine—and adjusted the hyperparameters to investigate various outcomes. The model's accuracy on the training set maximizes under the specified hyperparameters. Finally, the accuracy of the two models on the training set was compared using the optimum parameter combination and 10-fold cross-validation. The neural network model was discovered to be marginally superior, but it also had an overfitting problem. Overall, both models worked well, and I suggest the support vector machine for patient health prediction based on the test set's accuracy, which showed that it was more accurate.