STATS 790
Assignment 4

Doudou Jin – 400174871

4/9/23

```r
library(readr)
library(tidymodels)
library(vcd)
library(gridExtra)
library(ggplot2)
library(dplyr)
library(randomForest)
library(caret)
```

Dataset

Descriptions

The early stage diabetes risk prediction dataset("Early Stage Diabetes Risk Prediction Dataset." 2020) will be used in this assignment, which can be downloaded from the UCI Machine Learning Repository. This dataset includes the diagnosis outcomes (Positive or Negative) of early stage diabetes.

The dataset was collected through direct questionnaires from patients of Sylhet Diabetes Hospital in Sylhet, Bangladesh, and was given from a publication *Likelihood Prediction of Diabetes at Early Stage Using Data Mining Techniques* by Islam, MM Faniqul, et al (Islam et al. 2020). Three machine learning algorithms were employed in this study to predict the diabetes risk, including Naive Bayes (NB), Logistic Regression, and Random Forest (RF). The researchers found that Random Forest was the best classification method with the highest overall accuracy in predicting the likelihood of diabetes at an early stage.

The dataset contains 520 patients and a total of 17 attributes. There are 2 demographic features (i.e. Age, Gender) and 14 clinical features which include both common and less common sign symptoms. All the clinical features are binary, with values of either No or Yes. The Gender is also binary with Male or Female as its values, and the Age is the only predictor that is continuous. The response variable, *class*, is also binary, with values of either Positive or Negative.

```r
# import dataset

diabetes <- read_csv("diabetes_data_upload.csv")
```

```r
# correct the class of each variable

for (i in names(diabetes)){
  if (i == "Age"){
    diabetes[[i]] <- diabetes[[i]]
  }
  else {
    diabetes[[i]] <- as.factor(diabetes[[i]])
  }
}
```

```r
# change column names to correct the column name format

colnames(diabetes) <- c("Age", "Gender", "Polyuria", "Polydipsia"
                        , "suddenWeightLoss", "weakness", "Polyphagia"
                        , "GenitalThrush", "visualBlurring", "Itching"
                        , "Irritability", "delayedHealing", "partialParesis"
                        , "muscleStiffness", "Alopecia", "Obesity", "class")
```

Explanatory analysis

```r
# check missing values

for (col in names(diabetes)) {
  num_missing <- sum(is.na(diabetes[[col]]))
  cat(col, "has", num_missing, "missing values.\n")
}
```

```
Age has 0 missing values.
Gender has 0 missing values.
Polyuria has 0 missing values.
Polydipsia has 0 missing values.
suddenWeightLoss has 0 missing values.
weakness has 0 missing values.
Polyphagia has 0 missing values.
GenitalThrush has 0 missing values.
visualBlurring has 0 missing values.
Itching has 0 missing values.
Irritability has 0 missing values.
delayedHealing has 0 missing values.
partialParesis has 0 missing values.
muscleStiffness has 0 missing values.
Alopecia has 0 missing values.
Obesity has 0 missing values.
class has 0 missing values.
```

```r
# proportion of positive class and nagetive class

diabetes %>%
  group_by(class) %>%
  summarise(proportions = n()/520)
```

```
# A tibble: 2 x 2
  class    proportions
  <fct>          <dbl>
1 Negative       0.385
2 Positive       0.615
```

```r
barplot(table(diabetes$class))
```

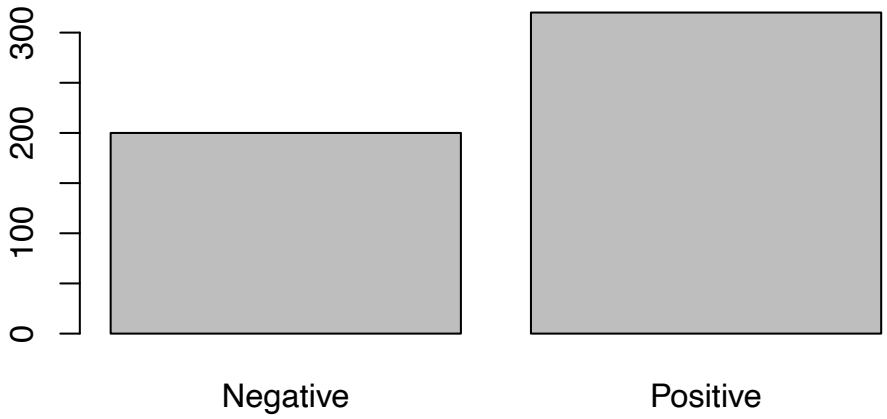

Figure 1: Bar Plot for Class

There are no missing values in this dataset, so there is no need for data imputation. The bar plot shows that the number of Positive classes is higher than that of Negative classes, with an approximate ratio of 8:5. Based on this ratio, we do not consider this dataset to be imbalanced.

```
# create mosaic plots to investigate associations between each predictor
# and response variable

vcd::mosaic(xtabs(~Gender+class, data = diabetes),shade=TRUE)

f1 <- grid.grab()

vcd::mosaic(xtabs(~Polyuria+class, data = diabetes),shade=TRUE)

f2 <- grid.grab()

vcd::mosaic(xtabs(~Polydipsia+class, data = diabetes),shade=TRUE)

f3 <- grid.grab()

vcd::mosaic(xtabs(~suddenWeightLoss+class, data = diabetes),shade=TRUE)

f4 <- grid.grab()
```

```r
vcd::mosaic(xtabs(~weakness+class, data = diabetes),shade=TRUE)
```

```r
f5 <- grid.grab()
```

```r
vcd::mosaic(xtabs(~Polyphagia+class, data = diabetes),shade=TRUE)
```

```r
f6 <- grid.grab()
```

```r
vcd::mosaic(xtabs(~GenitalThrush+class, data = diabetes),shade=TRUE)
```

```r
f7 <- grid.grab()
```

```r
vcd::mosaic(xtabs(~visualBlurring+class, data = diabetes),shade=TRUE)
```

```r
f8 <- grid.grab()
```

```r
vcd::mosaic(xtabs(~Itching+class, data = diabetes),shade=TRUE)
```

```r
f9 <- grid.grab()
```

```r
vcd::mosaic(xtabs(~Irritability+class, data = diabetes),shade=TRUE)
```

```r
f10 <- grid.grab()
```

```r
vcd::mosaic(xtabs(~delayedHealing+class, data = diabetes),shade=TRUE)
```

```r
f11 <- grid.grab()
```

```r
vcd::mosaic(xtabs(~partialParesis+class, data = diabetes),shade=TRUE)
```

```r
f12 <- grid.grab()
```

```r
vcd::mosaic(xtabs(~muscleStiffness+class, data = diabetes),shade=TRUE)
```

```
f13 <- grid.grab()
```

```
vcd::mosaic(xtabs(~Alopecia+class, data = diabetes),shade=TRUE)
```

```
f14 <- grid.grab()
```

```
vcd::mosaic(xtabs(~Obesity+class, data = diabetes),shade=TRUE)
```

```
f15 <- grid.grab()
```

```
grid.arrange(f1,f2, ncol = 2, nrow = 1)
```

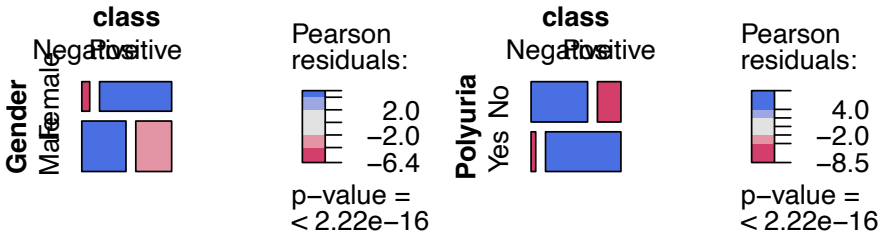

Figure 2: Mosaic Plots for Gender and Polyuria

```
grid.arrange(f3,f4, ncol = 2, nrow = 1)
```

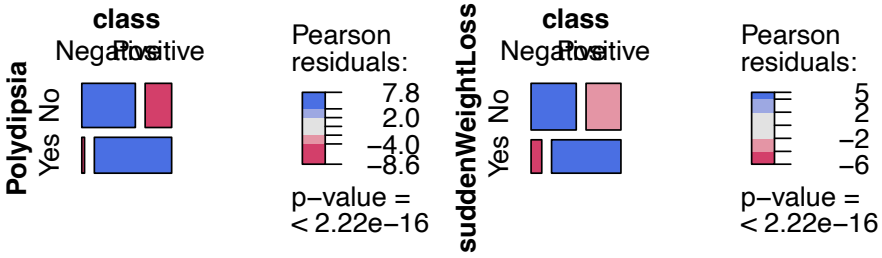

Figure 3: Mosaic Plots for Polydipsia and suddenWeightLoss

```
grid.arrange(f5,f6, ncol = 2, nrow = 1)
```

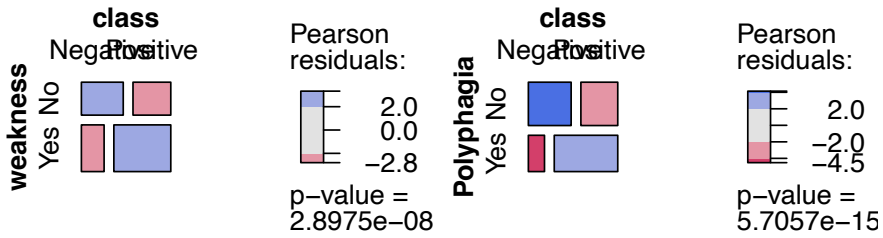

Figure 4: Mosaic Plots for weakness and Polyphagia

```
grid.arrange(f7,f8, ncol = 2, nrow = 1)
```

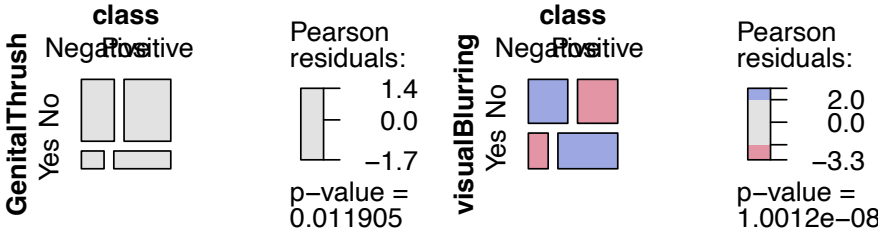

Figure 5: Mosaic Plots for GenitalThrush and visualBlurring

```
grid.arrange(f9,f10, ncol = 2)
```

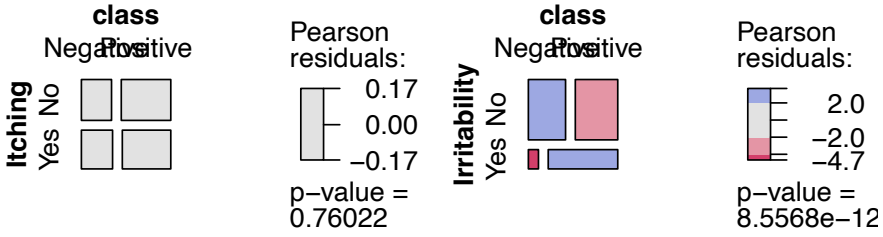

Figure 6: Mosaic Plots for Itching and Irritability

```
grid.arrange(f11,f12, ncol = 2, nrow = 1)
```

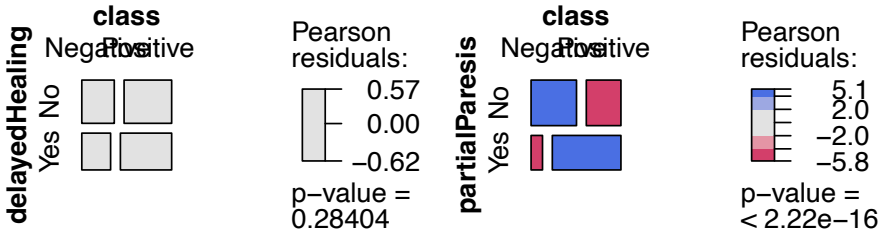

Figure 7: Mosaic Plots for delayedHealing and partialParesis

```
grid.arrange(f13,f14, ncol = 2, nrow = 1)
```

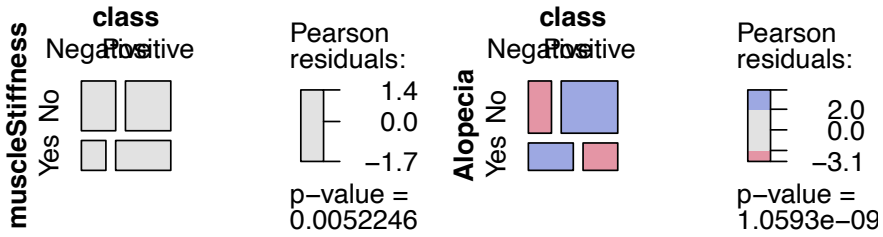

Figure 8: Mosaic Plots for muscleStiffness and Alopecia

```
grid.arrange(f15, ncol = 1, nrow = 1)
```

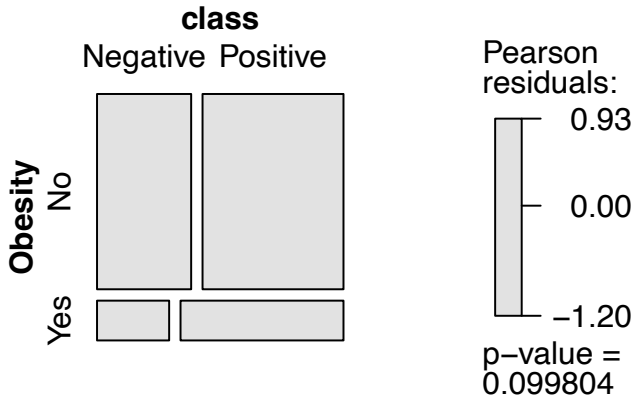

Figure 9: Mosaic Plot for Obesity

```
# create density plot to investigate associations between Age and class


d1 <- ggplot(data = diabetes, aes(x = Age, fill = class)) +
        geom_density(alpha = 0.5) +
        theme_minimal() +
        labs(title = "Density Plot for Age")
d1
```

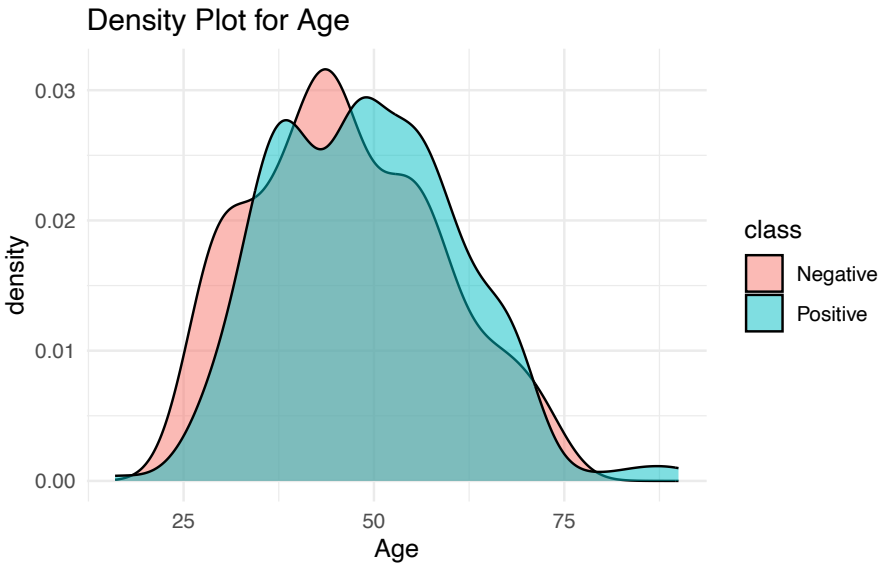

Figure 10: Density Plot for Age

```
diabetes %>% group_by(class) %>% summarize(mean = mean(Age))
```

```
# A tibble: 2 x 2
  class     mean
  <fct>    <dbl>
1 Negative  46.4
2 Positive  49.1
```

```
# perform Mann-Whitney test

wilcox.test(Age~class, data = diabetes)
```

```
    Wilcoxon rank sum test with continuity correction

data:  Age by class
W = 27834, p-value = 0.0124
alternative hypothesis: true location shift is not equal to 0
```

Based on the above mosaic plots, it is evident that the predictors *Gender*, *Polyuria*, *Polydipsia*, *suddenWeightLoss*, *Polyphagia*, and *partialParesis* have strong associations with the response variable,

with corresponding p-values that are extremely small. This suggests that these variables may play important roles in the prediction process. By considering both the density plot and Mann-Whitney test for Age, we can observe that the distributions of the Positive class and Negative class by age are different, implying that age is also a statistically significant variable for prediction. Additionally, the mean age of the Positive class is slightly higher than that of the Negative class. It is reasonable to assume that the likelihood of diabetes increases with age.

split dataset

```
# split dataset in training set and test set, with a ratio of 7:3
set.seed(1)

train <- diabetes %>% mutate(index=1:nrow(diabetes)) %>%
  mutate(n=n()) %>%
  sample_frac(size=0.7, weight=n) %>%
  ungroup()

train_index <- train$index
test <- diabetes[-train_index, ]
train <- train[,1:17]
```

To preserve the underlying patterns, we split the dataset into a training set and a test set with a ratio of 7:3. This will allow the algorithm to learn the patterns and avoid overfitting.

Feature Selection

```
set.seed(1)
diabetesFS <- rfcv(trainx = train[,1:16], trainy = train$class, cv.fold = 5
                   , scale = "log", step = 0.8,
                   mtry=function(p) max(1, floor(sqrt(p))), recursive=FALSE)
with(diabetesFS, plot(n.var, error.cv, type="o", lwd=2
                      , xlab ="Number of Variables", ylab = "CV Error"))
```

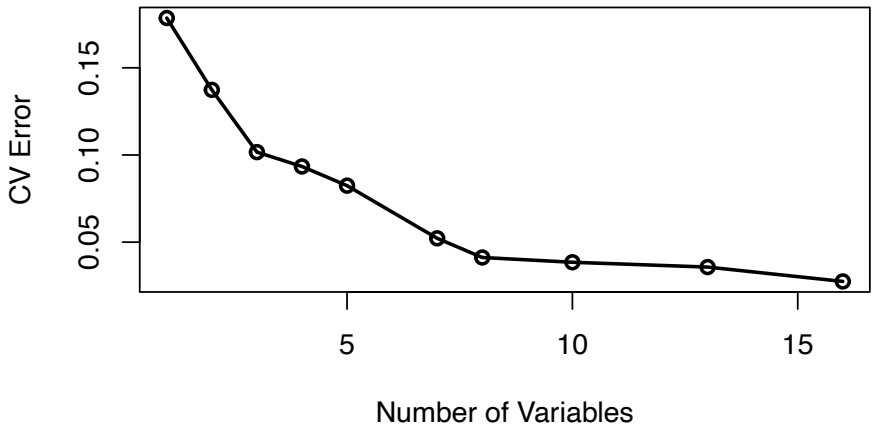

Figure 11: Error Rate by Number of Variables

Feature selection is an important process in model fitting. Removing redundant predictors can reduce noise and improve prediction performance. In this assignment, we have employed the Random Forest Cross-Validation method for feature selection. Five-fold cross-validation has been used to train the model and assess prediction performance. The number of predictors has been reduced based on their ranking of variable importance.

Based on the plot, we can observe that the error rates decrease rapidly from 1 variable to 8 variables, and then decrease steadily from 10 variables onwards, reaching the lowest error rates at 16 variables. Consequently, the error rate is the lowest when all predictors are utilized. Based this finding, we decided to keep all predictors.

Random Forest

Random Forest was selected for this assignment due to its excellent prediction performance in the publication. Random Forest(Breiman 2001) is a supervised machine learning algorithm that ensembles a collection of decision trees and obtains the prediction performance by averaging the performance of each decision tree. Since the decision trees are built by drawing bootstrap samples and fitting the out-of-bag data, the error rate is also referred to as the OOB error rate.

Random Forest is suitable for both continuous and categorical predictors and generally performs well on classification problems. Therefore, it is an appropriate method for this dataset.

Note that the Random Forest does not require scaling or one-hot encoding as it is a tree-based method. Additionally, we should be aware that both predictors, *Polyuria* and *Polydipsia*, have very small proportions of the Negative class when they both have sign symptoms. However, both variables are important predictors, and we do not consider dropping or lumping this category.

Tune the model

```
# tune the model

set.seed(1)

diabetesTune <- tuneRF(x = train[,1:16], y = train$class, mtryStart = 2
                       , ntreeTry = 500, stepFactor = 2, improve = 0.05
                       , plot = TRUE, doBest = TRUE)
```

```
mtry = 2  OOB error = 4.67%
Searching left ...
mtry = 1    OOB error = 5.22%
-0.1176471 0.05
Searching right ...
mtry = 4    OOB error = 2.47%
0.4705882 0.05
mtry = 8    OOB error = 3.02%
-0.2222222 0.05
```

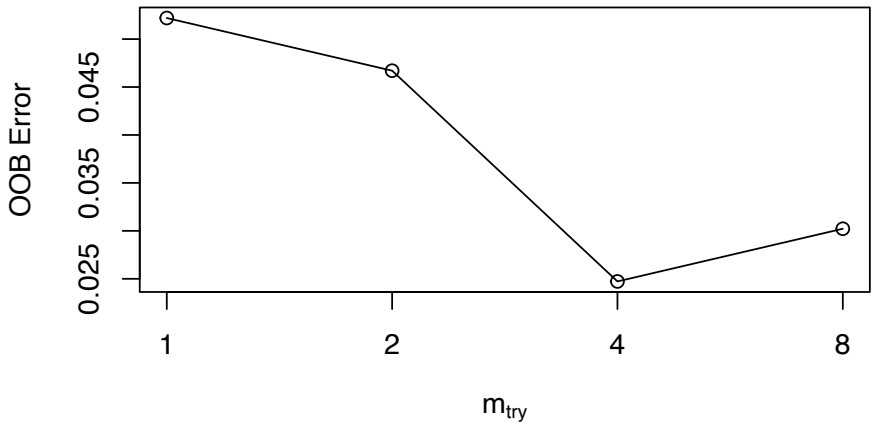

Figure 12: OOB Error Rate by Values of mtry

Firstly, the hyperparameter *mtry* is tuned. The optimal value of *mtry* is 4, which corresponds to the lowest OOB error rate. The optimal value of *mtry* is determined by fitting different values of *mtry* to the model and estimating the corresponding OOB error rate. In this case, the loss function is the OOB error rate, which ranges from 0 to 1.

```
set.seed(1)
diabetesRF1 <- randomForest(class~., data = train, ntree=1000, mtry = 4)

plot(diabetesRF1)
```

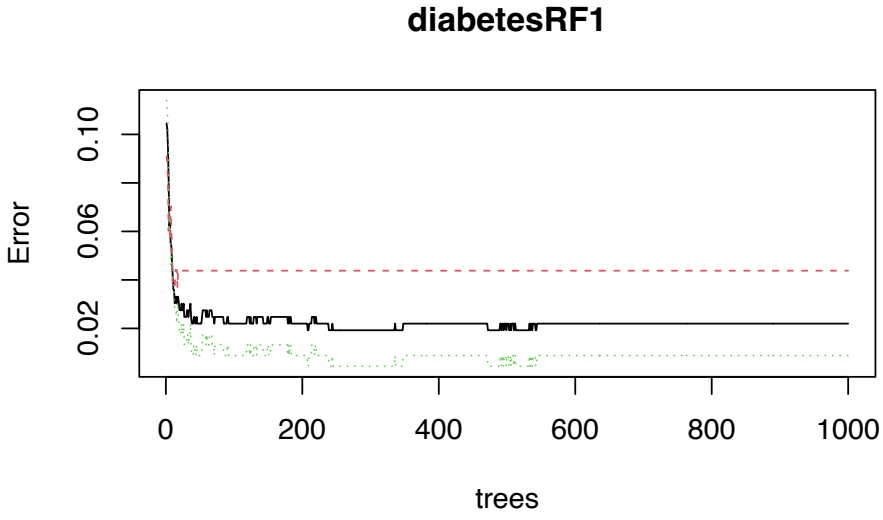

Figure 13: OOB Error Rate by Number of Trees, ntree = 1000

```r
set.seed(1)
diabetesRF <- randomForest(class~., data = train, ntree=520, mtry = 4)
diabetesRF
```

```
Call:
 randomForest(formula = class ~ ., data = train, ntree = 520,      mtry = 4)
               Type of random forest: classification
                     Number of trees: 520
No. of variables tried at each split: 4

        OOB estimate of  error rate: 1.92%
Confusion matrix:
         Negative Positive class.error
Negative      131        6 0.043795620
Positive        1      226 0.004405286
```

```r
plot(diabetesRF)
```

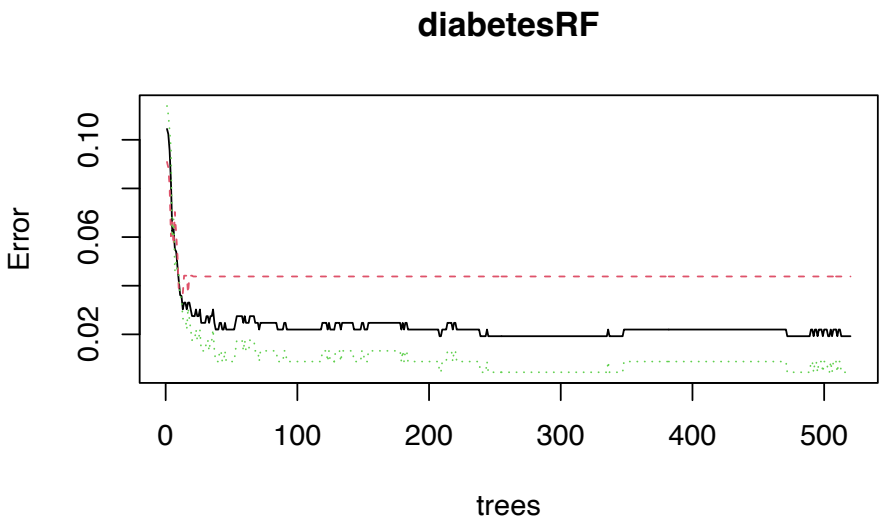

Figure 14: OOB Error Rate by Number of Trees, ntree = 520

Next, the number of trees (*ntree*) is tuned. By ensembling 1000 decision trees, we can observe that the OOB error rate remains constant after around 550 trees. Therefore, we can reduce the number

of trees to 550 and determine that the optimal number of trees is 520, which corresponds to the lowest OOB error rate.

Fitting the Model and Prediction

```r
diabetesRFpred <- predict(diabetesRF, test)
confusionMatrix(diabetesRFpred, test$class, positive = "Positive")
```

```
Confusion Matrix and Statistics

          Reference
Prediction Negative Positive
  Negative       62        4
  Positive        1       89

               Accuracy : 0.9679
                 95% CI : (0.9268, 0.9895)
    No Information Rate : 0.5962
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9339

 Mcnemar's Test P-Value : 0.3711

            Sensitivity : 0.9570
            Specificity : 0.9841
         Pos Pred Value : 0.9889
         Neg Pred Value : 0.9394
             Prevalence : 0.5962
         Detection Rate : 0.5705
   Detection Prevalence : 0.5769
      Balanced Accuracy : 0.9706
```

```
       'Positive' Class : Positive
```

After fitting the model with the optimal values of *mtry* and *ntree* and predicting using the test set, the prediction performance is shown above. The overall accuracy is 96.79%. Four patients in the Negative class are classified as Positive, and one patient in the Positive class is classified as Negative. As our goal is to predict the diabetes risk in the early stage, it is more important to predict the Positive class accurately. The sensitivity measures how well the model classifies the Positive class. For this model, the sensitivity is slightly lower than the specificity, while the positive predicted value is a little higher than the negative predicted value. This indicates that the model performs slightly better in identifying the Negative class.

Variable Importance

```
imp <- importance(diabetesRF)
imp
```

```
                  MeanDecreaseGini
Age                       15.712464
Gender                    17.573274
Polyuria                  41.569246
Polydipsia                28.976620
suddenWeightLoss          10.313176
weakness                   3.429669
Polyphagia                 3.899259
GenitalThrush              4.322178
visualBlurring             4.097421
Itching                    4.175915
Irritability               6.801331
delayedHealing             4.908596
partialParesis             6.992195
```

```
muscleStiffness          3.817705
Alopecia                 6.613571
Obesity                  2.816828
```

```
varImpPlot(diabetesRF)
```

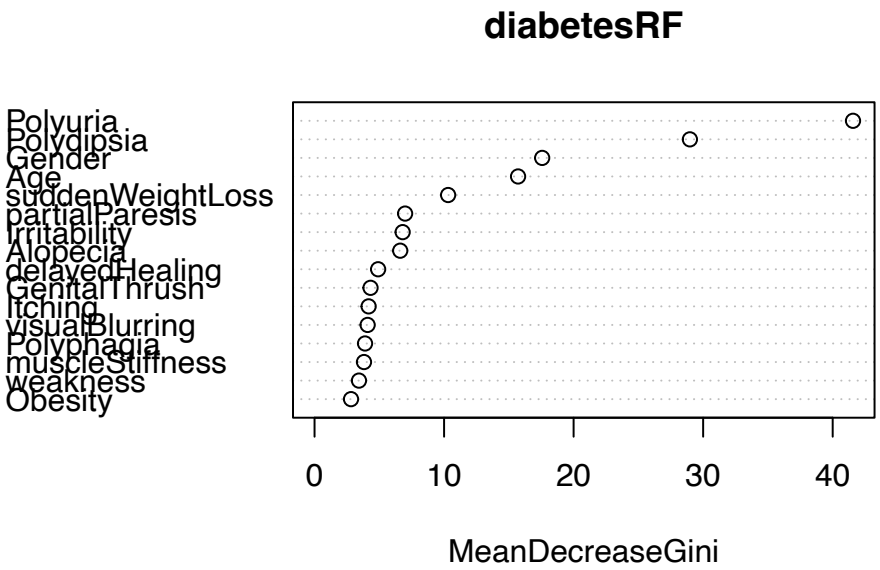

Figure 15: Variable Importance Plot

The mean decrease in Gini is a measure that assesses how well the predictors affect the node impurity in a decision tree. The node impurity is a measure that reflects how well the decision tree is split. The higher the mean decrease in Gini, the more important the predictors are for splitting the decision tree, as they contribute more to reducing the node impurity.

From the plot above, we observe that the predictors, *Polyuria*, *Polydipsia*, *Gender*, *Age* and *suddenWeightLoss* play important roles in Random Forest model, which confirmed the results we observed in exploratory analysis.

Partial Dependent Plot

```
impvar <- rownames(imp)[order(imp[, 1], decreasing=TRUE)]
op <- par(mfrow=c(2, 3))
for (i in 1:6) {
    partialPlot(x = diabetesRF, pred.data = as.data.frame(train)
                , x.var = impvar[i], xlab = impvar[i],
                main=paste("Partial Dependence on", impvar[i])
                , which.class = "Positive")
}
```

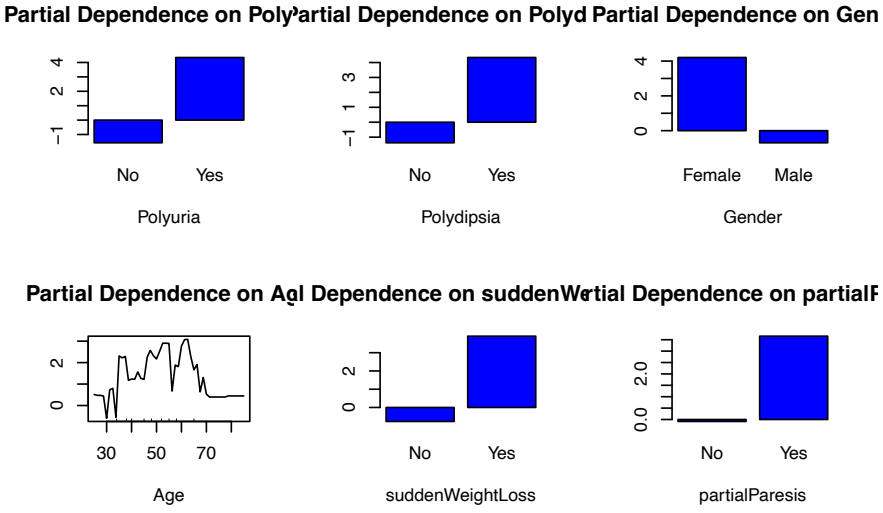

Figure 16: Partial Dependence Plots for Each Predictors (1)

```
par(op)
```

```
op <- par(mfrow=c(2, 3))
for (i in 7:12) {
    partialPlot(x = diabetesRF, pred.data = as.data.frame(train)
                , x.var = impvar[i], xlab = impvar[i],
                main=paste("Partial Dependence on", impvar[i]))
}
```

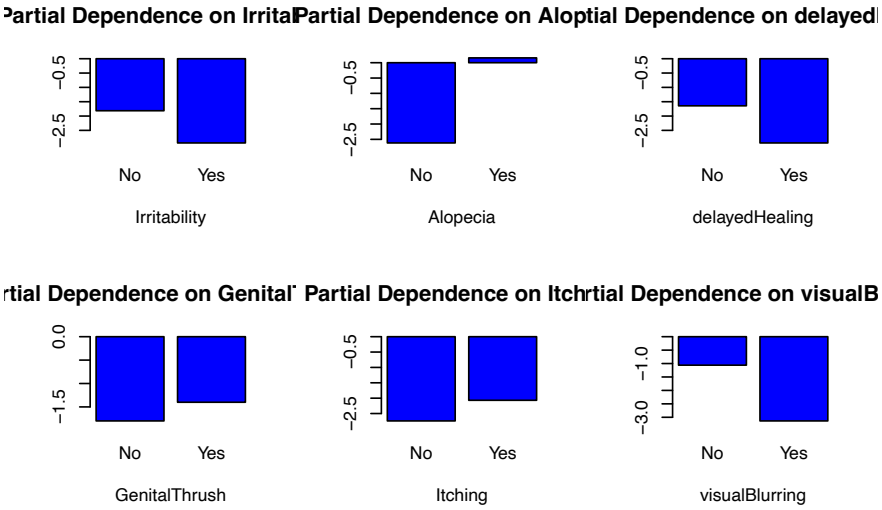

Figure 17: Partial Dependence Plots for Each Predictors (2)

```
par(op)
```

```
op <- par(mfrow=c(2, 3))
for (i in 13:16) {
    partialPlot(x = diabetesRF, pred.data = as.data.frame(train)
                , x.var = impvar[i], xlab = impvar[i],
                main=paste("Partial Dependence on", impvar[i]))
}
par(op)
```

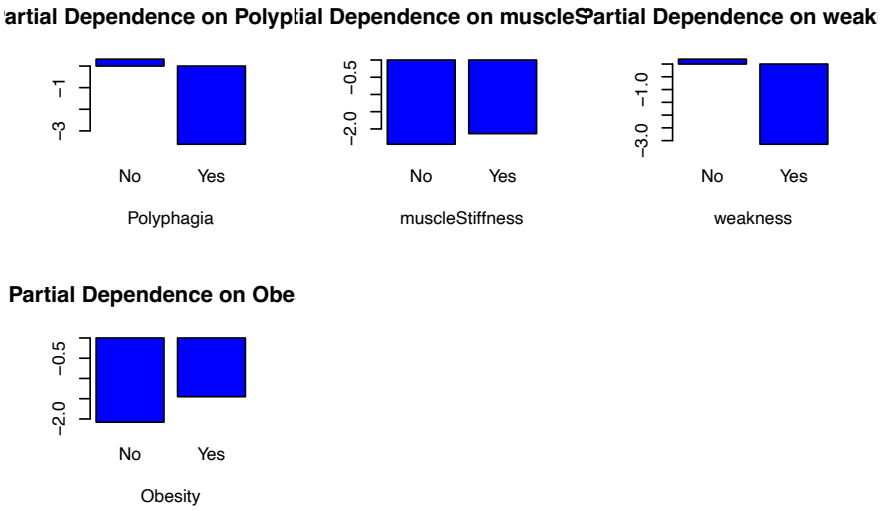

Figure 18: Partial Dependence Plots for Each Predictors (3)

The partial dependence plots show the marginal effect of each predictor on predicting Positive class, and the plots are ranked by variable importance. There are several predictors with low variable importance but have a high marginal effect on predicting Positive class, including *weakness*, *Polyphagia* and *Alopecia*.

Q2.

(a) For MSE loss function:

$$L(y, f(x)) = \frac{1}{N}(y - f(x))^2$$

=> step (c): suppose the region R_{jm} has k entries

$$\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$

$$= \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{k} \frac{1}{k}(y_i - f_{m-1}(x_i) - \gamma)^2$$

$$\frac{\partial}{\partial \gamma}\left[\sum_{i=1}^{k} \frac{1}{k}(y_i - f_{m-1}(x_i) - \gamma)^2\right]$$

$$= -\frac{2}{k}\sum_{i=1}^{k} y_i + \frac{2}{k}\sum_{i=1}^{k} f_{m-1}(x_i) + \frac{2}{k} \cdot k\gamma$$

$$= 2\gamma - \frac{2}{k}\sum_{i=1}^{k}(y_i - f_{m-1}(x_i))$$

=> set it to 0,

$$2\gamma - \frac{2}{k}\sum_{i=1}^{k}(y_i - f_{m-1}(x_i)) = 0$$

$$\hat{\gamma}_{jm} = \frac{1}{k}\sum_{i=1}^{k}(y_i - f_{m-1}(x_i))$$

Figure 19: Q2.a

Hence, the optimal value of $\hat{\gamma}_{jm}$ for MSE loss function is the mean of residuals $(y_i - f_{m-1}(x_i))$ in the corresponding region R.

Figure 20: Q2.a

For binomial deviance loss function,

$$-l(y, f(x)) = \log\left(1 + e^{-2yf(x)}\right)$$

$$\Rightarrow \gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{im}} L(y_i, f_{m-1}(x_i) + \gamma)$$

$$= \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{im}} -\log\left(1 + e^{-2y_i f_{m-1}(x_i) - 2y_i \gamma}\right)$$

$$= \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{im}} -\log\left(1 + e^{-2y_i f_{m-1}(x_i)} \, e^{-2y_i \gamma}\right)$$

$$= \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{im}} -\log\left(1 + w_i^{(m)} e^{-2y_i \gamma}\right)$$

where $w_i^{(m)} = e^{-y_i f_{m-1}(x_i)}$

differentiate in terms of γ,

$$\frac{\partial}{\partial \gamma}\left[\sum_{x_i \in R_{im}} -\log\left(1 + w_i^{(m)} e^{-2y_i \gamma}\right)\right]$$

Figure 21: Q2.a

$$= \sum_{x_i \in Rim} - \frac{-2y_i \, w_i^{2(m)} \, e^{-2y_i\gamma}}{1 + w_i^{2(m)} \, e^{-2y_i\gamma}}$$

$$= \sum_{x_i \in Rim} (y_i) \frac{2 \, w_i^{2(m)} \, e^{-2y_i\gamma}}{1 + w_i^{2(m)} \, e^{-2y_i\gamma}}$$

$$= \sum_{x_i \in Rim,\, y_i = 1} \frac{2 \, w_i^{2(m)} \, e^{-2\gamma}}{1 + w_i^{2(m)} \, e^{-2\gamma}} - \sum_{x_i \in Rim,\, y_i = -1} \frac{2 \, w_i^{2(m)} \, e^{2\gamma}}{1 + w_i^{2(m)} \, e^{2\gamma}}$$

set it to 0,

$$\sum_{x_i \in Rim,\, y_i = 1} \frac{2 \, w_i^{2(m)} \, e^{-2\gamma}}{1 + w_i^{2(m)} \, e^{-2\gamma}} = \sum_{x_i \in Rim,\, y_i = -1} \frac{2 \, w_i^{2(m)} \, e^{2\gamma}}{1 + w_i^{2(m)} \, e^{2\gamma}}$$

Solve for γ,

$$\Rightarrow \hat{\gamma}_{jm} = \frac{1}{2} \log \left(\frac{\sum_{x_i \in Rim} w_i^{(m)} \, \mathbb{I}(y_i = 1)}{\sum_{x_i \in Rim} w_i^{(m)} \, \mathbb{I}(y_i = -1)} \right)$$

Figure 22: Q2.a

(b)

For L2 norm loss function:

$$L(y, f(x)) = (y - f(x))^2$$

$$g_i = \frac{\partial}{\partial \hat{y}^{(t-1)}} L(y_i, \hat{y}^{(t-1)})$$

$$= \frac{\partial}{\partial \hat{y}^{(t-1)}} (y_i - \hat{y}^{(t-1)})^2$$

$$= -2(y_i - \hat{y}^{(t-1)})$$

$$h_i = \frac{\partial}{\partial \hat{y}^{(t-1)}} g_i$$

$$= 2$$

$\Rightarrow$ optimal value of weight

Suppose there are k instances in leaf j

$$\hat{w}_j = -\frac{\sum_{i=1}^{k} g_i}{\sum_{i=1}^{k} h_i + \lambda}$$

$$= \frac{2 \sum_{i=1}^{k} y_i - \hat{y}_i^{(t-1)}}{2k + \lambda}$$

Figure 23: Q2.b

λ is the regularization parameter, if $\lambda = 0$ in this case, the optimal weights

$\hat{w}_j = \frac{1}{k} \sum_{i=1}^{k} y_i - \hat{y}_i^{(t-1)}$ is the mean of the residuals in leaf j.

Figure 24: Q2.b

For binomial deviance,

$$l(y, f(x)) = -\log\left(1 + e^{-2yf(x)}\right)$$

$$g_i = \frac{\partial}{\partial \hat{y}^{(t-1)}} L(y_i, \hat{y}^{(t-1)})$$

$$= \frac{\partial}{\partial \hat{y}_i^{(t-1)}} -\log\left(1 + e^{-2y_i \hat{y}_i^{(t-1)}}\right)$$

$$= -\frac{-2y_i e^{-2y_i \hat{y}_i^{(t-1)}}}{1 + e^{-2y_i \hat{y}^{(t-1)}}}$$

$$= \frac{2y_i e^{-2y_i \hat{y}_i^{(t-1)}}}{1 + e^{-2y_i \hat{y}_i^{(t-1)}}}$$

$$= \frac{2y_i}{1 + e^{2y_i \hat{y}_i^{(t-1)}}}$$

$$h_i = \frac{\partial}{\partial \hat{y}^{(t-1)}} g_i$$

$$= 2y_i \frac{\partial}{\partial \hat{y}^{(t-1)}} \frac{e^{-2y_i \hat{y}_i^{(t-1)}}}{1 + e^{-2y_i \hat{y}_i^{(t-1)}}}$$

$$= 2y_i \left(\frac{2y_i e^{-4y_i \hat{y}_i^{(t-1)}} - 2y_i e^{-2y_i \hat{y}_i^{(t-1)}} \left(1 + e^{-2y_i \hat{y}_i^{(t-1)}}\right)}{\left(1 + e^{-2y_i \hat{y}_i^{(t-1)}}\right)^2} \right)$$

Figure 25: Q2.b

$$= 2y_i \left(\frac{2y_i e^{-2y_i \hat{y}_i^{(t-1)}} \left(e^{-2y_i \hat{y}_i^{(t-1)}} - 1 - e^{-2y_i \hat{y}_i^{(t-1)}} \right)}{\left(1 + e^{-2y_i \hat{y}_i^{(t-1)}} \right)^2} \right)$$

$$= - \frac{4y_i^2 \, e^{-2y_i \hat{y}_i^{(t-1)}}}{\left(1 + e^{-2y_i \hat{y}_i^{(t-1)}} \right)^2}$$

$$= - \frac{4y_i^2 \, e^{2y_i \hat{y}_i^{(t-1)}}}{\left(1 + e^{2y_i \hat{y}_i^{(t-1)}} \right)^2}$$

$$\Rightarrow \quad \hat{w}_j = - \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

$$= \frac{\sum_{i \in I_j} \frac{2y_i}{1 + e^{2y_i \hat{y}_i^{(t-1)}}}}{\sum_{i \in I_j} \frac{4y_i^2 \, e^{2y_i \hat{y}_i^{(t-1)}}}{\left(1 + e^{2y_i \hat{y}_i^{(t-1)}} \right)^2} + \lambda}$$

Figure 26: Q2.b

Reference

Auguie, Baptiste. 2017. *gridExtra: Miscellaneous Functions for "Grid" Graphics*. Manual.

Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32. https://doi.org/10.1023/A:1010933404324.

Bujokas, Eligijus. 2022. "Gradient Boosting in Python from Scratch." *Medium*.

Chen, Tianqi, and Carlos Guestrin. 2016. "XGBoost: A Scalable Tree Boosting System." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–94. KDD '16. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2939672.2939785.

"Early Stage Diabetes Risk Prediction Dataset." 2020. UCI Machine Learning Repository.

Islam, M. M. Faniqul, Rahatara Ferdousi, Sadikur Rahman, and Humayra Yasmin Bushra. 2020. "Likelihood Prediction of Diabetes at Early Stage Using Data Mining Techniques." In *Computer Vision and Machine Intelligence in Medical Image Analysis*, edited by Mousumi Gupta, Debanjan Konar, Siddhartha Bhattacharyya, and Sambhunath Biswas, 113–25. Singapore: Springer Singapore.

Kuhn, Max. 2022. *Caret: Classification and Regression Training*. Manual.

Kuhn, Max, and Hadley Wickham. 2020. *Tidymodels: A Collection of Packages for Modeling and Machine Learning Using Tidyverse Principles.* Manual.

Liaw, Andy, and Matthew Wiener. 2002. "Classification and Regression by randomForest." *R News* 2 (3): 18–22.

Meyer, David, Achim Zeileis, and Kurt Hornik. 2006. "The Strucplot Framework: Visualizing Multi-Way Contingency Tables with Vcd." *Journal of Statistical Software* 17 (3): 1–48. https://doi.org/10.18637/jss.v017.i03.

———. 2023. *Vcd: Visualizing Categorical Data*. Manual.

Tibshirani, R., T. Hastie, and J. H. Friedman. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction : With 200 Full-color Illustrations*. Springer Series in Statistics. Springer.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. 2023. *Dplyr: A Grammar of Data Manipulation*. Manual.

Wickham, Hadley, Jim Hester, and Jennifer Bryan. 2023. *Readr: Read Rectangular Text Data*. Manual.

Zeileis, Achim, David Meyer, and Kurt Hornik. 2007. "Residual-Based Shadings for Visualizing (Conditional) Independence." *Journal of Computational and Graphical Statistics* 16 (3): 507–25. https://doi.org/10.1198/106186007X237856.