Data 622 Homework 2

Brandon Cunningham, Jean Jimenez, Chafiaa Nadour, Shri Tripathi

2024-11-02

Assignment Instructions

Based on the latest topics presented, choose a dataset of your choice and create a Decision Tree where you can solve a classification problem and predict the outcome of a particular feature or detail of the data used.

Switch variables* to generate 2 decision trees and compare the results. Create a random forest and analyze the results.

Based on real cases where desicion trees went wrong, and 'the bad & ugly' aspects of decision trees (https://decizone.com/blog/the-good-the-bad-the-ugly-of-using-decision-trees),,) how can you change this perception when using the decision tree you created to solve a real problem?

Pre-Work

Large Dataset (Diabetes Data)

For our large dataset, we will use the diabetes dataset from kaggle.

This dataset has 100k clinical records of diabetes for health analytic purposes.

Link to Dataset

Goal: For this dataset, we want to predict whether or not the patient will have diabetes. In other words, this is a classification problem where we have to correctly label data. We will do this using decision trees and a random forest

Importing:

```
diabetes_data=read.csv(url("https://raw.githubusercontent.com/sleepysloth12/DATA_622_HW01/refs/heads/ma
```

Exploratory Data Analysis

The column of interest, labeled `diabetes` is what we want to predict. It is an integer, 0 or 1, indicating if the patient has diabetes or not. In the current dataset, 91% of the patients have no diabetes and 8.5% of the patients have diabetes.

In order to build a predictive model, we must first go column by column and clean up the features a little bit to make this more accurate/applicable to healthcare data.

Data Cleaning

Year The first column is year. The dataset is timeseries data, collected from the years 2015-2022. However, each year has different numbers of observations. There is no way of knowing if this is longtitudnal data (one patient visited multiple year) due to the lack of unique patient identifier field. I think we can completely disregard and forget about this column.

```r
diabetes_data = diabetes_data %>%
  select(-year)
```

Gender Next is gender. Gender is pretty even split, with ~60% being female and ~40% being male. There is an insignificant amount of people that answered "other" (less than 1%).

I'm going ahead and going to filter out other. Also, I am going to change the label to`is_female` so the choice is binary.

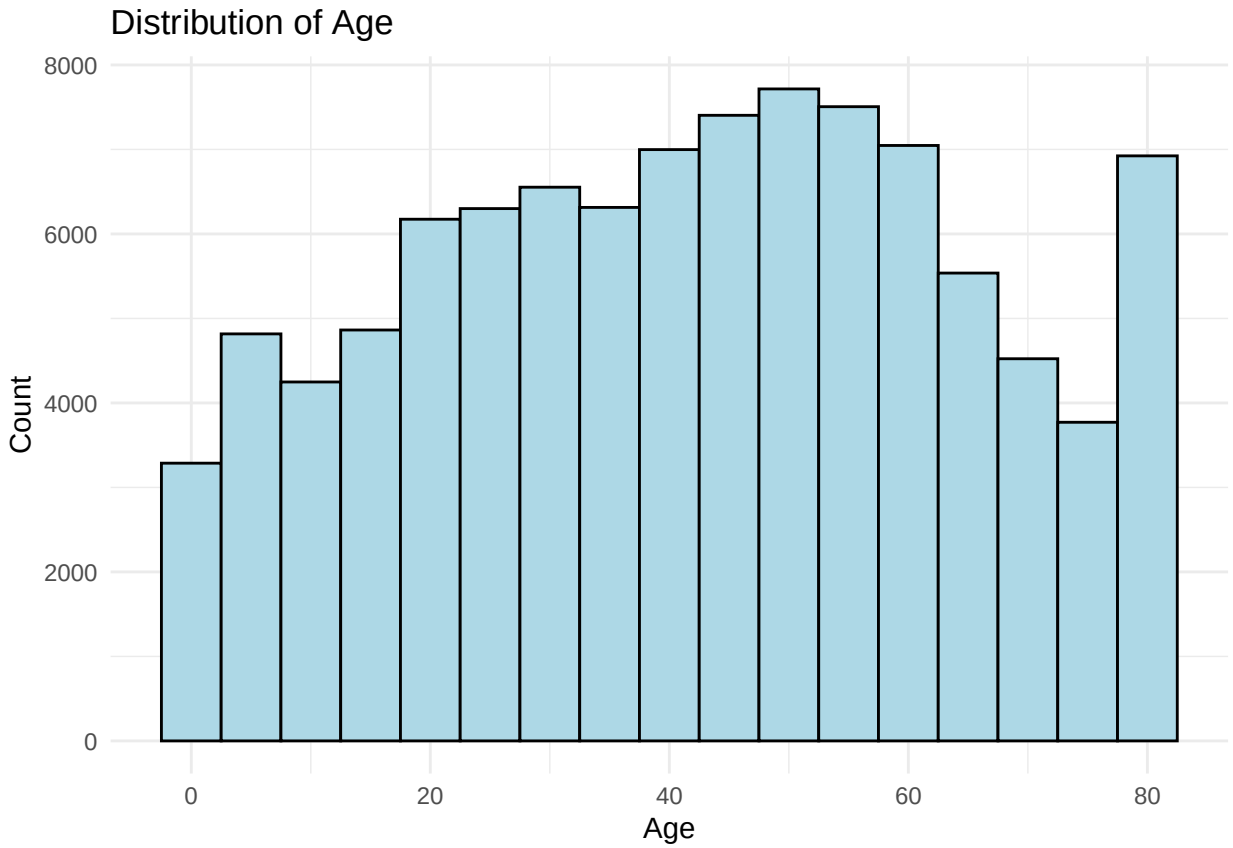
```r
diabetes_data = diabetes_data %>%
  filter(gender == "Female"| gender== "Male")%>%
  mutate(is_female=ifelse(gender == "Female",1,0))%>%
  select(-gender)
```

Age Next is age. Mean age is 41.9 years old, with a standard deviation of +/-22.5 years old.

Max age is 80.

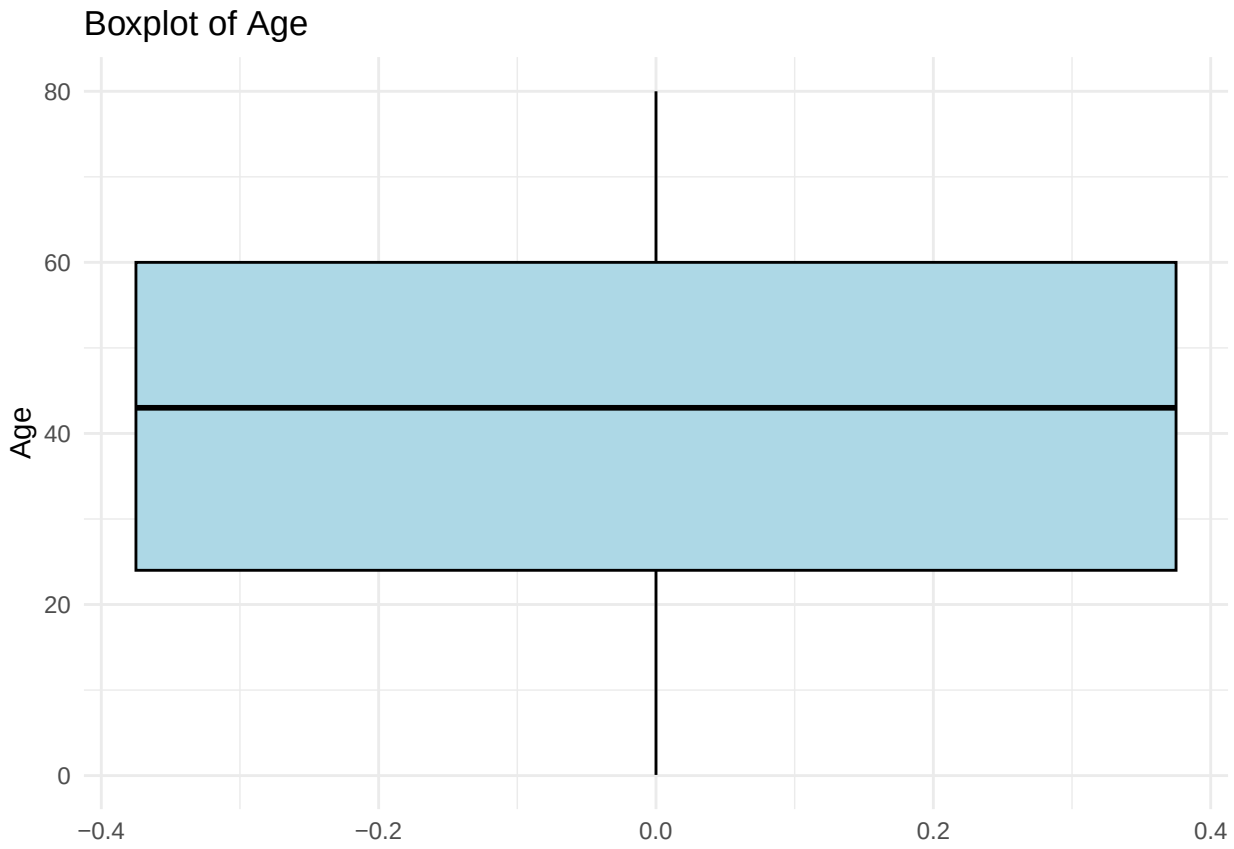
Minimum recorded age is 0.08. This might be an outlier. Therefore, lets visualize this distribution in both box plot and bar plot.

```r
ggplot(diabetes_data, aes(x = age)) +
  geom_histogram(binwidth = 5, color = "black", fill = "lightblue") +
  labs(title = "Distribution of Age", x = "Age", y = "Count") +
  theme_minimal()
```




```r
ggplot(diabetes_data, aes(y = age)) +
  geom_boxplot(fill = "lightblue", color = "black") +
```

```
  labs(title = "Boxplot of Age", y = "Age") +
  theme_minimal()
```



Seems like the minimum age is an outlier. In medical research, we tend to separate adult populations from pediatric populations so lets go ahead and do that here. Lets only look at 18+.

In terms of the age distribution, it looks relatively normal. Diabetes incidence seem to increase as you get closer to middle age, then decrease. There is a spike at 80 years old.

I am going to bin age/ convert it into different categories:

`is_young` = Age 18-35

`is_middle_age` = Age 36-64

`is_old` = Age 65+

```
diabetes_data = diabetes_data %>%
  filter(age>=18)%>%
  mutate(is_young=ifelse(age>=18 & age <=35, 1,0),
         is_middle_age=ifelse(age>35 & age<65 , 1 , 0),
         is_old=ifelse(age>65,1,0))%>%
  select(-age)
```

```
length(unique(diabetes_data$location))
```

State

```
## [1] 55
```

For the location column, there are 55 different locations, corresponding to the 50 different states and territory.

Location is important for diabetes prediction. Some areas are probably more likely to develop diabetes than others. Like age, I want to create categories and bin them based on the location. Then, will create dummy variables.

```
diabetes_data = diabetes_data %>%
  mutate(

    is_new_england = if_else(location %in% c("Connecticut", "Maine", "Massachusetts",
                                             "New Hampshire", "Rhode Island", "Vermont"), 1, 0),


    is_south = if_else(location %in% c("Alabama", "Arkansas", "Delaware", "Florida", "Georgia",
                                       "Kentucky", "Louisiana", "Maryland", "Mississippi",
                                       "North Carolina", "Oklahoma", "South Carolina",
                                       "Tennessee", "Texas", "Virginia", "West Virginia"), 1, 0),


    is_midwest = if_else(location %in% c("Illinois", "Indiana", "Iowa", "Kansas", "Michigan",
                                         "Minnesota", "Missouri", "Nebraska", "North Dakota",
                                         "Ohio", "South Dakota", "Wisconsin"), 1, 0),


    is_west = if_else(location %in% c("Alaska", "Arizona", "California", "Colorado", "Hawaii",
                                      "Idaho", "Montana", "Nevada", "New Mexico", "Oregon",
                                      "Utah", "Washington", "Wyoming"), 1, 0),


    is_northeast = if_else(location %in% c("New Jersey", "New York", "Pennsylvania"), 1, 0),


    is_territories = if_else(location %in% c("Guam", "Puerto Rico", "Virgin Islands",
                                             "District of Columbia", "United States"), 1, 0)
  )%>%
  select(-location)
```

Race, Ethnicity, Hypertension, & Heart Disease Race and ethnicity is already binned and with their individual dummy variables. Race and ethnicity are both factors that influence diabetes so will leave these columns untouched.

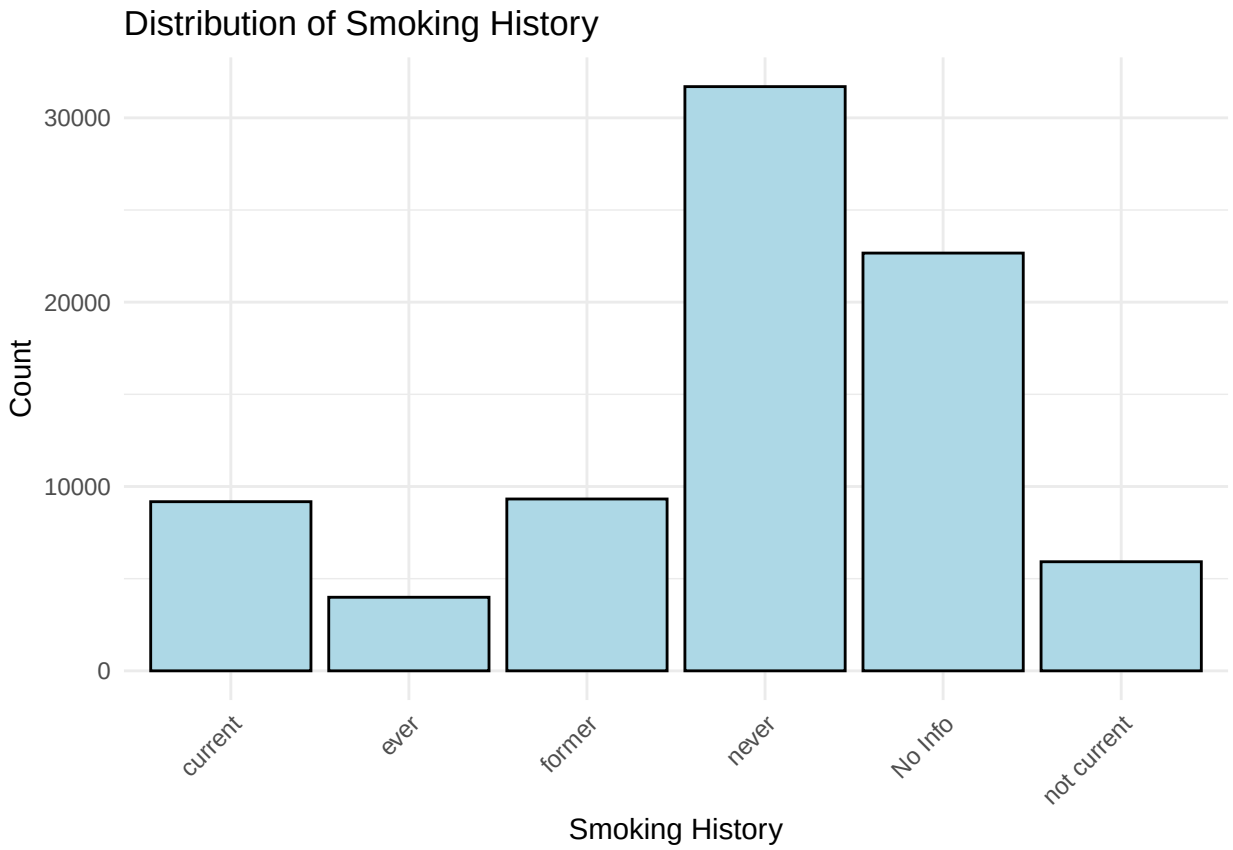
Same with the columns of hypertension and heart disease.

Smoking History There are currently 6 categories/ choices patients could respond when asked about smoking history:

```
unique(diabetes_data$smoking_history)
```

```
## [1] "never"       "not current" "current"     "No Info"     "ever"
## [6] "former"
```

```
ggplot(diabetes_data, aes(x = smoking_history)) +
  geom_bar(fill = "lightblue", color = "black") +
  labs(title = "Distribution of Smoking History", x = "Smoking History", y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



The biggest group is Never smoked accounting for 35% of the data.

There is a category, 'ever' which is 'Never' mislabeled. Will fix this. Once combined, never smoked will account for40% of the data.

The second biggest is 'No info' with near 35% of the data. Since the people in 'No info' may or may not be smokers, if we leave this category in it might make our predictions inaccurate. We want to capture how smoking can influence diabetes, therefore we ill remove this group.

Also, the 'not current' and 'former' group can be combined.

```
diabetes_data = diabetes_data %>%
  filter(smoking_history!="No Info")%>%
  mutate(never_smoked=ifelse(smoking_history %in% c("ever","never"),1,0),
         former_smoker=ifelse(smoking_history %in% c("former","not current"),1,0),
         current_smoker=ifelse(smoking_history=="current",1,0)) %>%
  select(-smoking_history)
```

Biomarker Columns The distribution of BMI is normal. It is numeric and continuous. We are leaving this as is.

The hbA1c_level biomarker, although numeric, has 18 unique values. In healthcare, this biomarker is usually used to determine diabetes. We will bin this biomarker for the following categories:

A1c < 5.7% –> Normal A1C

A1c between 5.7-6.4 % –> PreDiabetes

A1C over 6.5% –> diabetes

Although, correlation analysis is needed. There might be multicollinearity between these biomarker variables.

I say this because blood glucose variable and A1c directly related to each other.

Actually going to remove blood glucose because having that and A1C is repetitive/ multicollinearity.

```
diabetes_data = diabetes_data %>%
  mutate(normal_a1c=ifelse(hbA1c_level<5.7,1,0),
         prediabetic_a1c=ifelse(hbA1c_level>=5.7 & hbA1c_level <= 6.4,1,0),
         diabetic_a1c=ifelse(hbA1c_level>6.4,1,0))%>%
  select(-c(hbA1c_level,blood_glucose_level))
```

Building the Models

Now that our dataset is clean, we can discuss what model we want to use.

The target variable to predict is diabetes (binary choice whether or not patient will have diabetes).

For the sake of the assignment and to answer the questions, we will be training two decision trees and one random forrest.

Correlation Matrix

Before building the models, I want to run a correlation matrix to look for multicolinearity

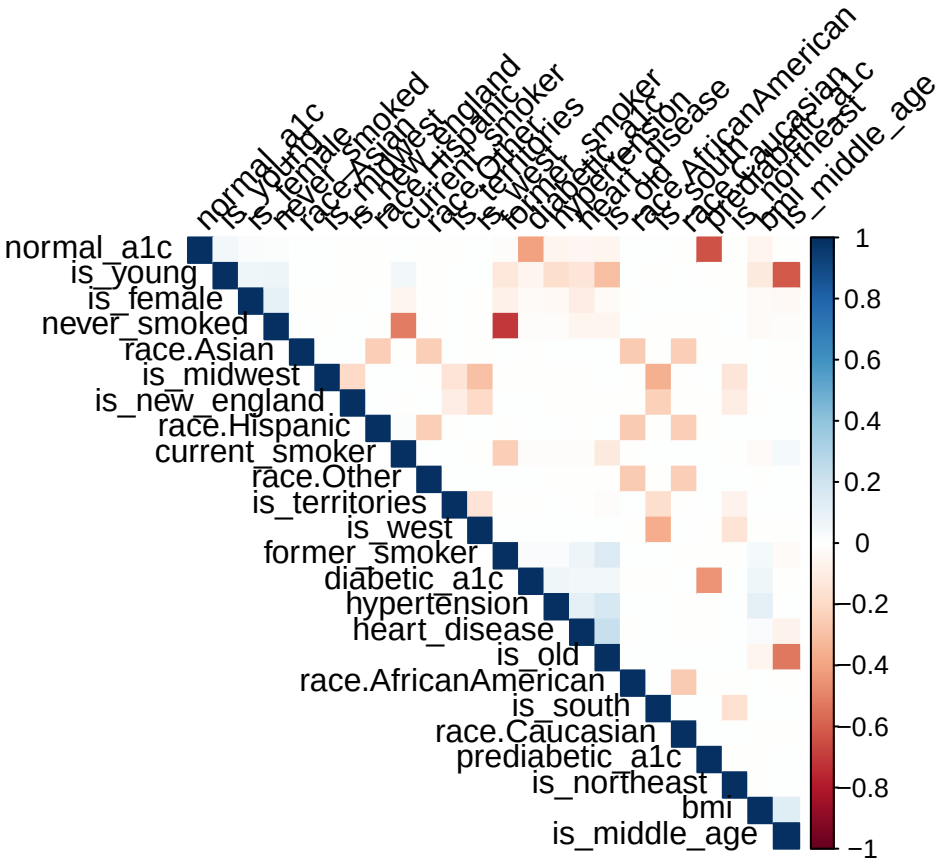
```
predictors <- diabetes_data %>%
  select(-diabetes)

cor_matrix <- cor(predictors)

high_cor <- findCorrelation(cor_matrix, cutoff = 0.7, verbose = TRUE)
```

```
## Compare row 19  and column  20 with corr  0.705
##   Means:  0.07 vs 0.049 so flagging column 19
## All correlations <= 0.7
```

```
cat("Highly correlated variables:", paste(names(predictors)[high_cor], collapse = ", "), "\n")
```

```
## Highly correlated variables: never_smoked
```

```
corrplot(cor_matrix, method = "color", type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45)
```



There is some multicollinearity

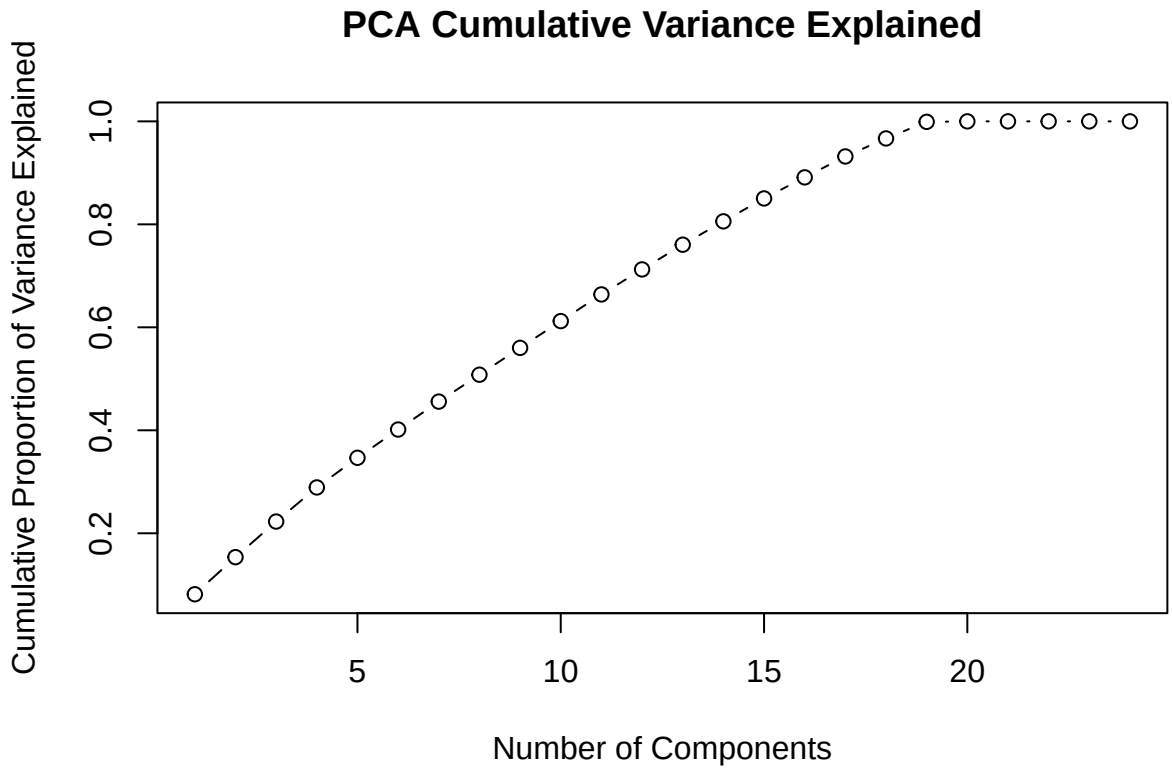
Principal Component Analysis

Conducting a PCA to determine the important components

```
pca_result <- prcomp(predictors, scale. = TRUE)
summary(pca_result)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5    PC6     PC7
## Standard deviation     1.39898 1.3155 1.28787 1.26107 1.17547 1.1478 1.14083
## Proportion of Variance 0.08155 0.0721 0.06911 0.06626 0.05757 0.0549 0.05423
## Cumulative Proportion  0.08155 0.1537 0.22276 0.28902 0.34659 0.4015 0.45572
##                           PC8     PC9    PC10    PC11    PC12    PC13   PC14
## Standard deviation     1.11981 1.11834 1.11722 1.11510 1.07939 1.07351 1.0438
## Proportion of Variance 0.05225 0.05211 0.05201 0.05181 0.04855 0.04802 0.0454
## Cumulative Proportion  0.50797 0.56008 0.61209 0.66390 0.71244 0.76046 0.8059
##                          PC15    PC16    PC17    PC18    PC19    PC20
## Standard deviation     1.03300 0.99101 0.98754 0.91645 0.87923 0.14869
## Proportion of Variance 0.04446 0.04092 0.04063 0.03499 0.03221 0.00092
## Cumulative Proportion  0.85032 0.89124 0.93187 0.96687 0.99908 1.00000
##                            PC21     PC22      PC23      PC24
## Standard deviation     2.199e-13 2.56e-14 2.422e-14 1.844e-14
## Proportion of Variance 0.000e+00 0.00e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.00e+00 1.000e+00 1.000e+00
```

```r
plot(cumsum(pca_result$sdev^2 / sum(pca_result$sdev^2)),
     type = "b",
     xlab = "Number of Components",
     ylab = "Cumulative Proportion of Variance Explained",
     main = "PCA Cumulative Variance Explained")
```



Our first principal component only accounts for about 8.16% of the total variance. That's not a lot. It means no single factor dominates in predicting diabetes. This makes sense given the complex nature of the disease and the variety of factors we've included in our dataset.

We need 11 components to explain about 66% of the variance, and it takes 19 to get to nearly 100%. Looking at our cumulative variance plot, we can see this gradual climb. The fact that we need so many components to explain most of the variance suggests we shouldn't try to oversimplify our model. Most of our variables are contributing unique information about diabetes risk.

While we don't see extreme multicollinearity, there is some correlation among our variables. We can explain about 85% of the variance with 15 components, which is fewer than our original variables.

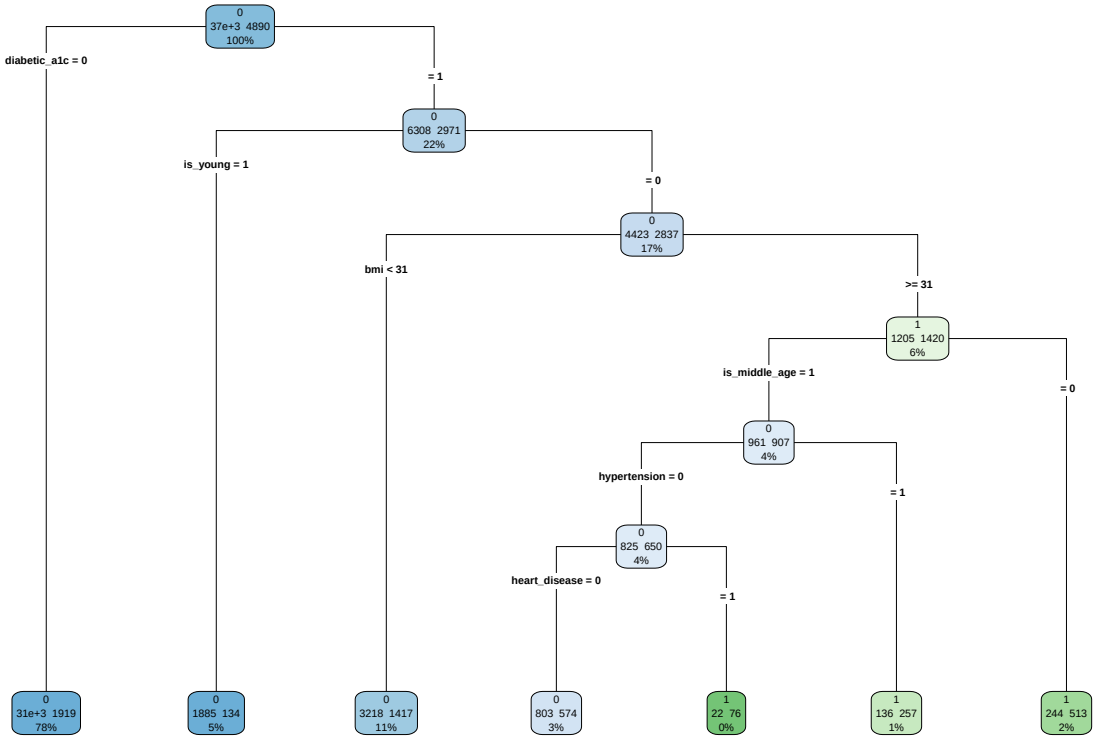
```r
set.seed(622)


train_split_idx=createDataPartition(diabetes_data$diabetes, p=0.7, list=FALSE)

train_diab = diabetes_data[train_split_idx,]
test_diab = diabetes_data[-train_split_idx,]

control <- trainControl(method = "cv", number = 5)
metric <- "Accuracy"
```

Models

Decision Tree 1

For this tree we are throwing everything into the decision tree and

```
set.seed(622)
fit_tree <- rpart(diabetes ~ ., method = 'class', data = train_diab)

rpart.plot(fit_tree, type = 4, extra = 101)
```



The decision tree model was trained on 42,073 observations with diabetes as the target variable. The most important variable for prediction was diabetic_a1c (importance score of 61), followed by is_young (20) and bmi (11). Other variables like is_middle_age, is_old, hypertension, and heart_disease had relatively lower importance scores.

At the root node (Node 1), about 11.6% of patients had diabetes (4,890 out of 42,073). The first major split occurs based on diabetic_a1c < 0.5, which creates two branches:

Left Branch (Node 2):

- Contains 32,794 patients (78% of total)
- Only 5.9% have diabetes in this group
- This node is a terminal node, predicting no diabetes

Right Branch (Node 3):

- Contains 9,279 patients (22% of total)
- 32% have diabetes
- Further splits based on age and BMI

The right branch continues to split based on the is_young variable, creating Node 6 (2,019 patients, 6.6% diabetic) and Node 7 (7,260 patients, 39% diabetic). Node 7 then splits on BMI < 30.585, indicating that higher BMI values are associated with increased diabetes risk.

```
printcp(fit_tree)
```

```
##
## Classification tree:
## rpart(formula = diabetes ~ ., data = train_diab, method = "class")
##
## Variables actually used in tree construction:
## [1] bmi           diabetic_a1c  heart_disease hypertension  is_middle_age
## [6] is_young
##
## Root node error: 4890/42073 = 0.11623
##
## n= 42073
##
##         CP nsplit rel error  xerror     xstd
## 1 0.014656      0   1.00000 1.00000 0.013444
## 2 0.011043      5   0.92025 0.92004 0.012963
## 3 0.010000      6   0.90920 0.91513 0.012932
```

```
summary(fit_tree)
```

```
## Call:
## rpart(formula = diabetes ~ ., data = train_diab, method = "class")
##   n= 42073
##
##           CP nsplit rel error    xerror       xstd
## 1 0.01465576      0 1.0000000 1.0000000 0.01344361
## 2 0.01104294      5 0.9202454 0.9200409 0.01296257
## 3 0.01000000      6 0.9092025 0.9151329 0.01293208
##
## Variable importance
##  diabetic_a1c      is_young           bmi is_middle_age        is_old
##            61            20            11             2             2
##  hypertension heart_disease
##             2             1
##
## Node number 1: 42073 observations,    complexity param=0.01465576
##   predicted class=0  expected loss=0.1162266  P(node) =1
##     class counts: 37183  4890
##    probabilities: 0.884 0.116
##   left son=2 (32794 obs) right son=3 (9279 obs)
##   Primary splits:
##       diabetic_a1c < 0.5    to the left,  improve=990.4332, (0 missing)
##       normal_a1c   < 0.5    to the right, improve=656.6893, (0 missing)
##       hypertension < 0.5    to the left,  improve=302.2151, (0 missing)
##       is_young     < 0.5    to the right, improve=291.6968, (0 missing)
##       is_old       < 0.5    to the left,  improve=290.4417, (0 missing)
##   Surrogate splits:
##       bmi < 70.255 to the left,  agree=0.78, adj=0.001, (0 split)
##
## Node number 2: 32794 observations
```

```
##   predicted class=0  expected loss=0.0585168  P(node) =0.7794548
##     class counts: 30875  1919
##    probabilities: 0.941 0.059
##
## Node number 3: 9279 observations,    complexity param=0.01465576
##   predicted class=0  expected loss=0.3201854  P(node) =0.2205452
##     class counts:  6308  2971
##    probabilities: 0.680 0.320
##   left son=6 (2019 obs) right son=7 (7260 obs)
##   Primary splits:
##       is_young      < 0.5    to the right, improve=332.4822, (0 missing)
##       is_old        < 0.5    to the left,  improve=248.8659, (0 missing)
##       bmi           < 30.595 to the left,  improve=243.0985, (0 missing)
##       hypertension  < 0.5    to the left,  improve=222.8035, (0 missing)
##       heart_disease < 0.5    to the left,  improve=169.4133, (0 missing)
##
## Node number 6: 2019 observations
##   predicted class=0  expected loss=0.06636949  P(node) =0.04798802
##     class counts:  1885   134
##    probabilities: 0.934 0.066
##
## Node number 7: 7260 observations,    complexity param=0.01465576
##   predicted class=0  expected loss=0.3907713  P(node) =0.1725572
##     class counts:  4423  2837
##    probabilities: 0.609 0.391
##   left son=14 (4635 obs) right son=15 (2625 obs)
##   Primary splits:
##       bmi           < 30.585 to the left,  improve=185.4711, (0 missing)
##       hypertension  < 0.5    to the left,  improve=132.9508, (0 missing)
##       is_middle_age < 0.5    to the right, improve=126.4846, (0 missing)
##       is_old        < 0.5    to the left,  improve=114.9070, (0 missing)
##       heart_disease < 0.5    to the left,  improve=108.2956, (0 missing)
##
## Node number 14: 4635 observations
##   predicted class=0  expected loss=0.3057174  P(node) =0.1101657
##     class counts:  3218  1417
##    probabilities: 0.694 0.306
##
## Node number 15: 2625 observations,    complexity param=0.01465576
##   predicted class=1  expected loss=0.4590476  P(node) =0.06239156
##     class counts:  1205  1420
##    probabilities: 0.459 0.541
##   left son=30 (1868 obs) right son=31 (757 obs)
##   Primary splits:
##       is_middle_age < 0.5    to the right, improve=39.77034, (0 missing)
##       hypertension  < 0.5    to the left,  improve=33.33725, (0 missing)
##       heart_disease < 0.5    to the left,  improve=33.00677, (0 missing)
##       is_old        < 0.5    to the left,  improve=32.55981, (0 missing)
##       bmi           < 37.695 to the left,  improve=25.26904, (0 missing)
##   Surrogate splits:
##       is_old        < 0.5    to the left,  agree=0.971, adj=0.900, (0 split)
##       heart_disease < 0.5    to the left,  agree=0.712, adj=0.003, (0 split)
##
## Node number 30: 1868 observations,    complexity param=0.01465576
```

```
##   predicted class=0  expected loss=0.485546  P(node) =0.04439902
##     class counts:   961   907
##    probabilities: 0.514 0.486
##   left son=60 (1475 obs) right son=61 (393 obs)
##   Primary splits:
##       hypertension  < 0.5    to the left,  improve=28.228070, (0 missing)
##       heart_disease < 0.5    to the left,  improve=25.989720, (0 missing)
##       bmi           < 37.695 to the left,  improve=17.918840, (0 missing)
##       is_female     < 0.5    to the right, improve= 5.547072, (0 missing)
##       is_midwest    < 0.5    to the left,  improve= 2.477221, (0 missing)
##
## Node number 31: 757 observations
##   predicted class=1  expected loss=0.322325  P(node) =0.01799254
##     class counts:   244   513
##    probabilities: 0.322 0.678
##
## Node number 60: 1475 observations,    complexity param=0.01104294
##   predicted class=0  expected loss=0.440678  P(node) =0.03505811
##     class counts:   825   650
##    probabilities: 0.559 0.441
##   left son=120 (1377 obs) right son=121 (98 obs)
##   Primary splits:
##       heart_disease  < 0.5    to the left,  improve=23.537950, (0 missing)
##       bmi            < 37.365 to the left,  improve=16.420940, (0 missing)
##       is_female      < 0.5    to the right, improve= 3.990932, (0 missing)
##       is_midwest     < 0.5    to the left,  improve= 1.784707, (0 missing)
##       is_territories < 0.5    to the left,  improve= 1.739569, (0 missing)
##
## Node number 61: 393 observations
##   predicted class=1  expected loss=0.346056  P(node) =0.009340907
##     class counts:   136   257
##    probabilities: 0.346 0.654
##
## Node number 120: 1377 observations
##   predicted class=0  expected loss=0.4168482  P(node) =0.03272883
##     class counts:   803   574
##    probabilities: 0.583 0.417
##
## Node number 121: 98 observations
##   predicted class=1  expected loss=0.2244898  P(node) =0.002329285
##     class counts:    22    76
##    probabilities: 0.224 0.776
```

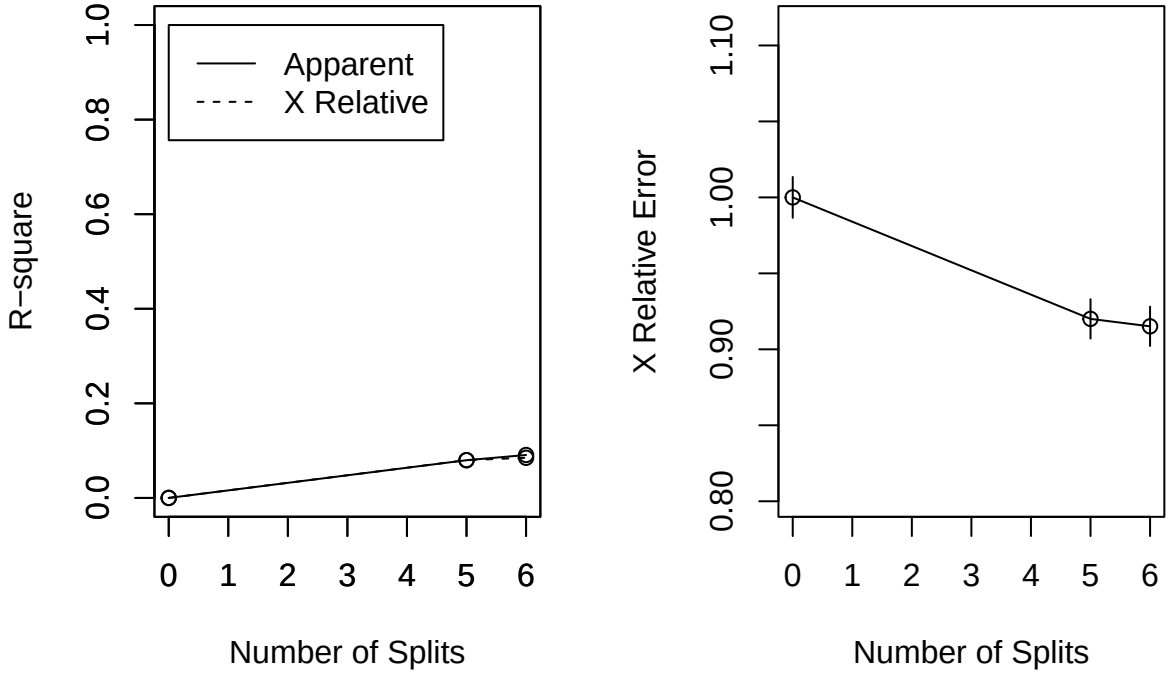
```
par(mfrow = c(1, 2))
rsq.rpart(fit_tree)
```



```
##
## Classification tree:
## rpart(formula = diabetes ~ ., data = train_diab, method = "class")
##
## Variables actually used in tree construction:
## [1] bmi           diabetic_a1c  heart_disease hypertension  is_middle_age
## [6] is_young
##
## Root node error: 4890/42073 = 0.11623
```

```
##
## n= 42073
##
##         CP nsplit rel error  xerror     xstd
## 1 0.014656      0   1.00000 1.00000 0.013444
## 2 0.011043      5   0.92025 0.92004 0.012963
## 3 0.010000      6   0.90920 0.91513 0.012932
```

```
## Warning in rsq.rpart(fit_tree): may not be applicable for this method
```




```
predictions_tree <- predict(fit_tree, newdata = test_diab)
head(predictions_tree)
```

```
##            0         1
## 1  0.9414832 0.0585168
## 2  0.9414832 0.0585168
## 9  0.5831518 0.4168482
## 11 0.5831518 0.4168482
## 12 0.9414832 0.0585168
## 14 0.9414832 0.0585168
```

```
accuracy <- mean(predictions_tree == test_diab$diabetes)
print(accuracy)
```

```
## [1] 0
```

The R-square plot shows very low values (around 0.1), indicating that the model explains only about 10% of the variance in the data. This might seem concerning, but for classification problems, especially with imbalanced classes like in this diabetes dataset, R-square isn't the most appropriate metric.

The X Relative Error plot shows a decreasing trend as the number of splits increases, suggesting that additional splits improve the model's performance, but only marginally after 5 splits. The error stabilizes around 0.91, indicating that further splits don't significantly improve prediction accuracy.

```
test_diab$diabetes <- as.factor(test_diab$diabetes)
predictions_tree <- predict(fit_tree, newdata = test_diab, type = "class")
conf_matrix <- confusionMatrix(predictions_tree, test_diab$diabetes)

print(conf_matrix$table)
```

```
##            Reference
## Prediction     0     1
##          0 15744  1750
##          1   169   367
```

Looking at the confusion matrix:

- True Negatives (TN) = 15,744 (correctly predicted non-diabetic patients)
- False Positives (FP) = 169 (incorrectly predicted diabetic)
- False Negatives (FN) = 1,750 (missed diabetic cases)
- True Positives (TP) = 367 (correctly predicted diabetic cases)

The model shows:

- High specificity (accurately identifying non-diabetic cases: 15,744/(15,744+169) 98.9%)
- Lower sensitivity (identifying diabetic cases: 367/(367+1,750) 17.3%)
- Overall accuracy of (15,744+367)/(15,744+169+1,750+367) 89.3%

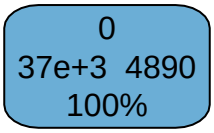
The model is conservative in predicting diabetes, with a very low false positive rate but a high false negative rate. This behavior might be influenced by the class imbalance in the original dataset. While the overall accuracy is good, the model's ability to identify actual diabetes cases needs improvement

Decision Tree 2

In this next model, it can't find any useful splits and ends up just having a base node of 0.

This is not a good model, but it is an interesting one.

```
set.seed(622)
train_diab2 <- train_diab[c("diabetes", "bmi", "never_smoked", "former_smoker", "is_young", "heart_disea
test_diab2 <- test_diab[c("diabetes", "bmi", "never_smoked", "former_smoker", "is_young", "heart_diseas

fit_tree2 <- rpart(diabetes ~ ., method = 'class', data = train_diab2)

rpart.plot(fit_tree2, type = 4, extra = 101)
```



The second decision tree, which was built using a reduced set of variables (BMI, smoking status, age indicator, heart disease, and hypertension), resulted in what's known as a "stump" - a tree with only one node and no splits.

```
printcp(fit_tree2)
```

```
##
## Classification tree:
## rpart(formula = diabetes ~ ., data = train_diab2, method = "class")
```

```
##
## Variables actually used in tree construction:
## character(0)
##
## Root node error: 4890/42073 = 0.11623
##
## n= 42073
##
##   CP nsplit rel error xerror xstd
## 1  0      0         1      0    0
```

```
summary(fit_tree2)
```

```
## Call:
## rpart(formula = diabetes ~ ., data = train_diab2, method = "class")
##   n= 42073
##
##   CP nsplit rel error xerror xstd
## 1  0      0         1      0    0
##
## Node number 1: 42073 observations
##   predicted class=0  expected loss=0.1162266  P(node) =1
##     class counts: 37183  4890
##    probabilities: 0.884 0.116
```

Looking at the statistics:

- **Total observations**: 42,073
- **Class distribution**: 37,183 non-diabetic (88.4%) vs 4,890 diabetic (11.6%)
- The model predicts class 0 (non-diabetic) for all cases with an expected loss of 0.1162
- Complexity parameter is 0, with no splits performed

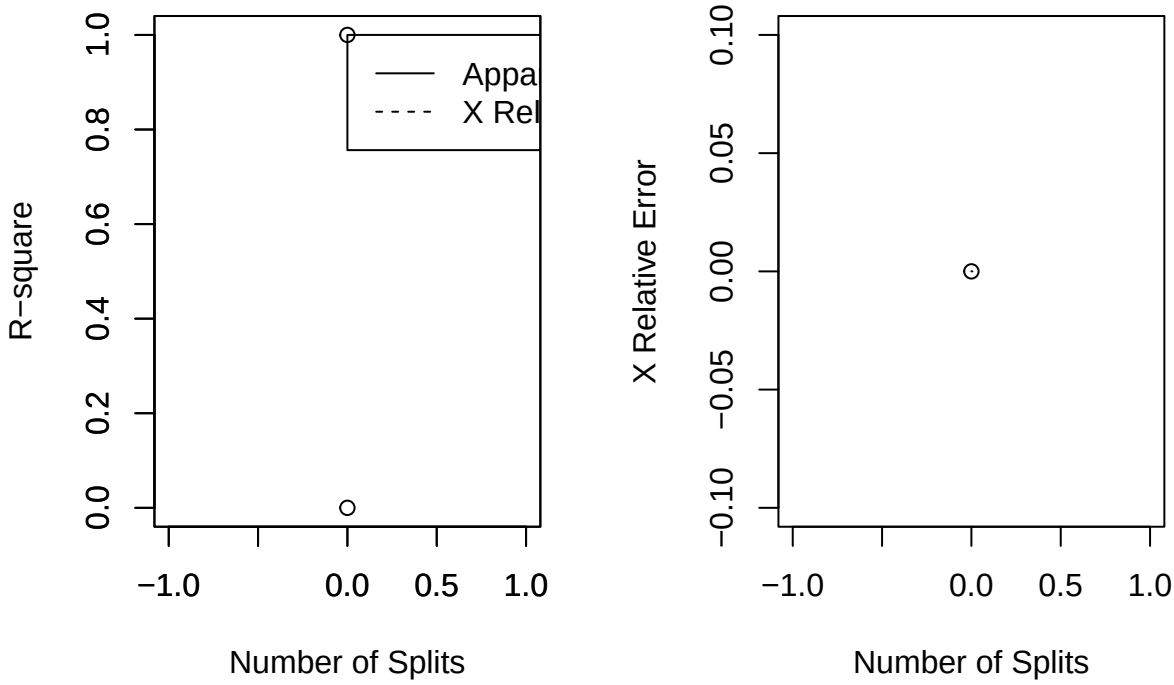
The model's behavior indicates that none of the included variables (BMI, smoking status, age, heart disease, and hypertension) provided enough information gain to justify creating any splits in the tree. This is a significant finding because it suggests that these variables alone, without the A1C levels that were crucial in the first tree, are not sufficient to make meaningful predictions about diabetes status.

This result aligns with medical knowledge that while factors like BMI, age, and cardiovascular health are risk factors for diabetes, they are not as directly diagnostic as blood markers like A1C levels.

```
par(mfrow = c(1, 2))
rsq.rpart(fit_tree2)
```

```
##
## Classification tree:
## rpart(formula = diabetes ~ ., data = train_diab2, method = "class")
##
## Variables actually used in tree construction:
## character(0)
##
## Root node error: 4890/42073 = 0.11623
##
## n= 42073
##
##   CP nsplit rel error xerror xstd
## 1  0      0         1      0    0
```

```
## Warning in rsq.rpart(fit_tree2): may not be applicable for this method
```




```r
predictions_tree <- predict(fit_tree2, newdata = test_diab2)
head(predictions_tree)
```

```
##              0         1
## [1,] 0.8837734 0.1162266
## [2,] 0.8837734 0.1162266
## [3,] 0.8837734 0.1162266
## [4,] 0.8837734 0.1162266
## [5,] 0.8837734 0.1162266
## [6,] 0.8837734 0.1162266
```

```r
accuracy <- mean(predictions_tree == test_diab2$diabetes)
print(accuracy)
```

```
## [1] 0
```

The plots show minimal variation because this is essentially a non-splitting tree. The R-square plot and X Relative Error plot both show single points near 0, which aligns with the tree's lack of splits and complexity.

Looking at the probability distributions:

- The model consistently predicts the same probabilities across all cases:
- 88.38% probability of no diabetes (class 0)
- 11.62% probability of diabetes (class 1)
- These probabilities exactly match the class distribution in the training data (4,890/42,073 = 11.62% diabetes prevalence)

```r
test_diab$diabetes <- as.factor(test_diab2$diabetes)
predictions_tree <- predict(fit_tree2, newdata = test_diab2, type = "class")
conf_matrix <- confusionMatrix(predictions_tree, test_diab2$diabetes)

print(conf_matrix$table)
```

```
##            Reference
## Prediction     0     1
##          0 15913  2117
##          1     0     0
```

The model made only "class 0" (no diabetes) predictions for all cases, which is reflected in the following metrics:

- True Negatives (TN) = 15,913 cases

These are patients correctly identified as not having diabetes

- False Negatives (FN) = 2,117 cases

These are patients who have diabetes but were incorrectly classified as non-diabetic

This is particularly concerning from a medical perspective as these are missed diagnoses

- True Positives (TP) = 0 cases

The model failed to correctly identify any diabetic patients

- False Positives (FP) = 0 cases

The model never predicted diabetes for any patient

Key performance metrics:

- Sensitivity (Recall) = 0% (0/2,117)
- Specificity = 100% (15,913/15,913)
- Accuracy = 88.2% (15,913/18,030)
- Precision = undefined (0/0)

This confusion matrix confirms that the model is essentially a naive classifier that always predicts "no diabetes" regardless of the input features. While it achieves a seemingly high accuracy of 88.2% due to class imbalance, it's clinically useless as it fails to identify any diabetic patients.

Comparing Decision Trees

The comparison of our two decision trees provides good insights into both the strengths and limitations of decision tree models in healthcare prediction. The first tree, which included A1C levels among its predictors, demonstrated reasonable predictive capability with an 89.3% accuracy rate and meaningful splits based on clinical markers. However, it still exhibited the "bad" aspects mentioned in the article; namely bias toward the majority class (non-diabetic) and limited sensitivity in detecting positive cases (only 17.3%). The second tree starkly illustrated the "ugly" aspects of decision trees when working with insufficient features. It devolved into a single-node stump with zero predictive value beyond majority class prediction. This addresses the article's concerns about subjectivity and complexity in decision trees.

To improve these models and counter the negative perception of decision trees in healthcare, we can try a few things:

1. Incorporate class balancing techniques to address the sensitivity issue
2. Use ensemble methods like random forests to improve robustness (which we will try out)

The stark difference between our two trees (one with biomarkers and one without) also emphasizes the article's point about the importance of proper feature selection and domain expertise in creating meaningful decision trees. While decision trees can be powerful tools in healthcare prediction, their successful implementation requires careful consideration of feature selection, class balance, and regular validation.

Random Forest Classifier

The random forest classifier aimed to predict the presence of diabetes using various health and demographic features in the dataset. After ensuring the target variable was treated as a categorical factor, the dataset was split into training and testing subsets, with 70% allocated to training. A simplified grid search was performed to identify the optimal number of features (`mtry`) to consider at each split, helping to balance model performance with runtime.

```
if (!require(randomForest)) install.packages("randomForest")
```

```
## Loading required package: randomForest
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

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

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

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(randomForest)
if (!require(caret)) install.packages("caret")
library(caret)


diabetes_data$diabetes <- as.factor(diabetes_data$diabetes)


set.seed(622)

sample_index <- sample(1:nrow(diabetes_data), size = 0.7 * nrow(diabetes_data))
train_data <- diabetes_data[sample_index, ]
test_data <- diabetes_data[-sample_index, ]

mtry_grid <- c(2, 4, 6)
best_mtry <- mtry_grid[1]
best_oob_error <- Inf

for (m in mtry_grid) {

  rf_temp <- randomForest(
    diabetes ~ .,
    data = train_data,
    ntree = 100,
    mtry = m,
    importance = TRUE
  )
```

```
  oob_error <- mean(rf_temp$err.rate[, "OOB"])

  if (!is.na(oob_error) && oob_error < best_oob_error) {
    best_oob_error <- oob_error
    best_mtry <- m
  }
}


rf_model <- randomForest(
  diabetes ~ .,
  data = train_data,
  ntree = 300,
  mtry = best_mtry,
  importance = TRUE
)


print(rf_model)
```

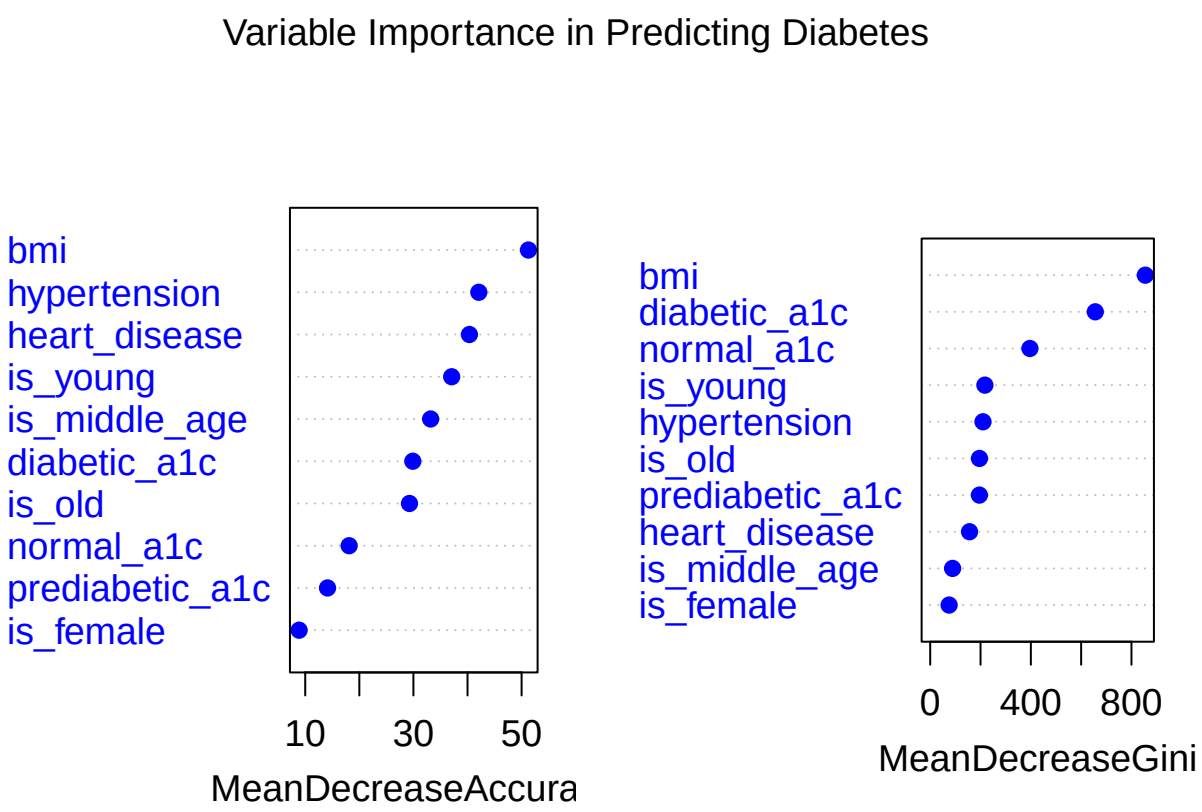
```
##
## Call:
##  randomForest(formula = diabetes ~ ., data = train_data, ntree = 300,      mtry = best_mtry, importa
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 10.45%
## Confusion matrix:
##       0    1 class.error
## 0 36568  615  0.01653982
## 1  3780 1109  0.77316425
```

```
rf_predictions <- predict(rf_model, test_data)
conf_matrix <- confusionMatrix(as.factor(rf_predictions), as.factor(test_data$diabetes))
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 15656  1643
##          1   257   475
##
##                Accuracy : 0.8946
##                  95% CI : (0.8901, 0.8991)
##     No Information Rate : 0.8825
##     P-Value [Acc > NIR] : 1.646e-07
##
##                   Kappa : 0.2905
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
##             Sensitivity : 0.9838
##             Specificity : 0.2243
##          Pos Pred Value : 0.9050
##          Neg Pred Value : 0.6489
##              Prevalence : 0.8825
##          Detection Rate : 0.8683
##    Detection Prevalence : 0.9594
##       Balanced Accuracy : 0.6041
##
##        'Positive' Class : 0
##
```

```r
varImpPlot(rf_model, main = "Variable Importance in Predicting Diabetes", n.var = 10, pch = 16, col = "
```



Using 300 trees with 4 variables per split, the Random Forest achieved an accuracy of 89.46%, marginally improving upon our first decision tree's performance of 89.3% and substantially outperforming the ineffective second "stump" tree.

The model's variable importance plot revealed BMI as the strongest predictor, followed by A1C-related variables and then demographic/comorbidity factors, aligning with clinical understanding of diabetes risk factors.

However, while the model showed excellent sensitivity (98.38%) for identifying non-diabetic cases, it struggled with specificity (22.43%) for diabetic cases, reflecting persistent challenges with class imbalance. This addresses the article's concerns about decision tree limitations while demonstrating how ensemble methods like Random Forests can mitigate some of these issues.

The differences between our three models shows both the potential and limitations of tree-based methods in medical prediction.The Random Forest's ability to capture complex variable interactions while maintaining reasonable accuracy suggests that ensemble methods, combined with careful feature selection and domain knowledge, can help overcome many of the traditional limitations of decision trees in healthcare applications.

However, this must be balanced against the increased complexity and reduced interpretability compared to simple decision trees, highlighting the ongoing trade-off between model performance and practicality in clinical settings.