Università degli Studi di Milano
Data Science and Economics (LM-91)

Statistical Learning

Early-Stage Diabetes Risk Prediction:
Which physiological symptoms to watch out for?

Shihab Hamati
Nov 3, 2022

Abstract

This paper explores the relationships of common physiological characteristics and symptoms to the incidence of diabetes. The dataset mainly consists mainly of binary features to indicate the presence or absence of symptoms. Four different supervised binary classification approaches are used to model the relationship between said physiological symptoms and diabetes. High levels of accuracy and sensitivity are achieved across all models which indicates the possibility to prioritize diabetes checkups in communities where more expensive blood tests and hospital accessibility may be limited by looking out for studied physiological symptoms.

Problem Statement

Diabetes is a chronic illness which is the result of the body not correctly producing or utilizing insulin, the blood glucose regulating hormone. According to WHO, 1 out of every 11 to 12 adults suffer from it. It was also considered the direct cause of death of 1.5 million in under-70 years old in 2019, in addition to half a million indirect kidney disease death and one-fifth of cardiovascular deaths. Sadly, there has been a 3% increase in diabetes related deaths between 2000 and 2019. This rate is much higher in lower-middle-income countries, at 13%.

With regular follow-up, proper medication, a healthy lifestyle, and a better diet, most people can lead a life of good quality. However, it is necessary to detect diabetes as early as possible to limit irreversible damage it may cause patients. Sadly, it is in lower income countries where diabetes deaths are increasing the fastest that early detection is hardest. This is due to many factors including costs of home blood glucose monitors and accessibility to healthcare.

Objective

The aim of this study is to model the relation between the incidence of diabetes and the presence of physiological symptoms. Community based detection can be performed by trained volunteers to assess the risk of diabetes in communities from observable physiological symptoms that do not require expensive blood tests or access to medical professionals. Such symptoms maybe be queried through paper, telephone, or online questionnaires that collect basic physiological profiles of respondents. Those whose answers indicate a higher risk of the presence of undiagnosed diabetes can then be prioritized to access limited medical attention in lower income communities.

To achieve this, an exploratory data analysis (EDA) is first performed on the features and response variable (diabetis status). Then, four supervised machine learning models are developed for the binary classification of the diabetic state of the patient. These models are logistic regression (LR), linear discriminant analysis (LDA), decision trees (DT), and random forests (RF).

Dataset

The dataset is sourced from UCI Machine Learning Repository (link). It contains 520 records. The records are direct questionnaire responses from patients at the Sylhet Diabetes Hospital in Bangladesh and approved by a doctor. It contains 17 columns: 1 numeric, 15 binary features (indicating presence or absence of symptoms), and 1 binary response variable (indicating the diabetic status). In this report, the features are classified into three general categories: overall characteristic traits, commonly named symptoms, and medically named symptoms.

Overall Characteristic Traits

- Age: *numeric*, representing the age of the patient in years
- Gender: *binary categorical*, male or female
- Obesity: *binary categorical*, indicates a BMI above 30 if true
- Sudden Weight Loss: *binary categorical*, indicate if the patient has experienced sudden and unintentional weight loss

Commonly Named Symptoms

All the following symptoms are intuitive and easy to detect by untrained persons.

- Weakness: *binary categorical*
- Muscle Stiffness: *binary categorical*
- Visual Blurring: *binary categorical*
- Itching: *binary categorical*
- Irritability: *binary categorical*
- Delayed Healing: *binary categorical*

Medically Named Symptoms

These symptoms are explained below, and their understanding may require health education of the population or community volunteers

- Polyuria: *binary categorical*, excessive urination either in frequency or volume
- Polydipsia: *binary categorical*, excessive thirst
- Polyphagia: *binary categorical*, excessive eating
- Paresis: *binary categorical*, muscular weakness, partial in this case
- Alopecia: *binary categorical*, bodily hair loss
- Genital Thrush: *binary categorical*, a fungal infection in the genitals

Exploratory Data Analysis

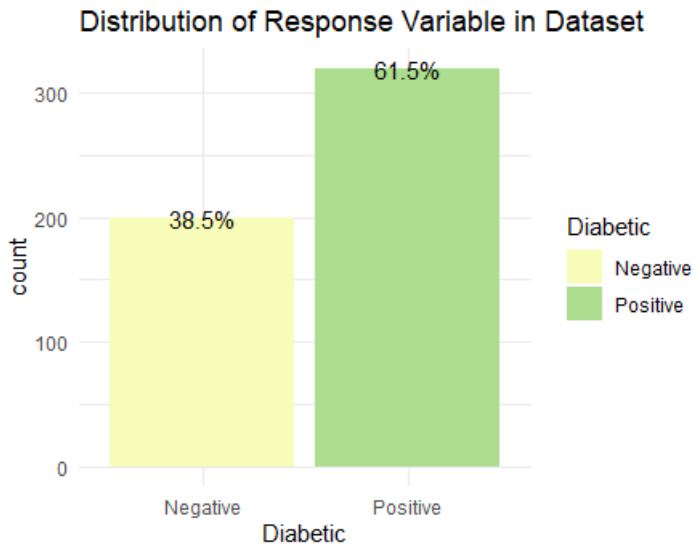
The dataset is in CSV format and does not suffer from any missing values or inadmissible responses. It consists of 520 records and 17 columns. Below is a transposed sample of the first 5 records.

Age	40	58	41	45	60
Gender	Male	Male	Male	Male	Male
Polyuria	No	No	Yes	No	Yes
Polydipsia	Yes	No	No	No	Yes
Sudden weight loss	No	No	No	Yes	Yes
Weakness	Yes	Yes	Yes	Yes	Yes
Polyphagia	No	No	Yes	Yes	Yes
Genital thrush	No	No	No	Yes	No
Visual blurring	No	Yes	No	No	Yes
Itching	Yes	No	Yes	Yes	Yes
Irritability	No	No	No	No	Yes
Delayed healing	Yes	No	Yes	Yes	Yes
Partial paresis	No	Yes	No	No	Yes
Muscle stiffness	Yes	No	Yes	No	Yes
Alopecia	Yes	Yes	Yes	No	Yes
Obesity	Yes	No	No	No	Yes
Class*	Positive	Positive	Positive	Positive	Positive

* Indicates diabetic status

Response Variable

The dataset contains mostly diabetic patients (61.5%). This does not reflect the actual population's diabetes incidence in Bangladesh which is at 12.8% (link). Thus, the dataset is imbalanced.



However, no balancing methods were used as this imbalance is not severe and does not impact the ability of the models to accurately capture and process the patterns and trends. The difference between sample and population distributions is not a factor in this analysis as both the training and test sets mimic the sample's target variable distributions. Nonetheless, it is not required to spend additional cost, time, and effort to gather further negative records as the models can be adjusted to account for the different population outcome distribution.

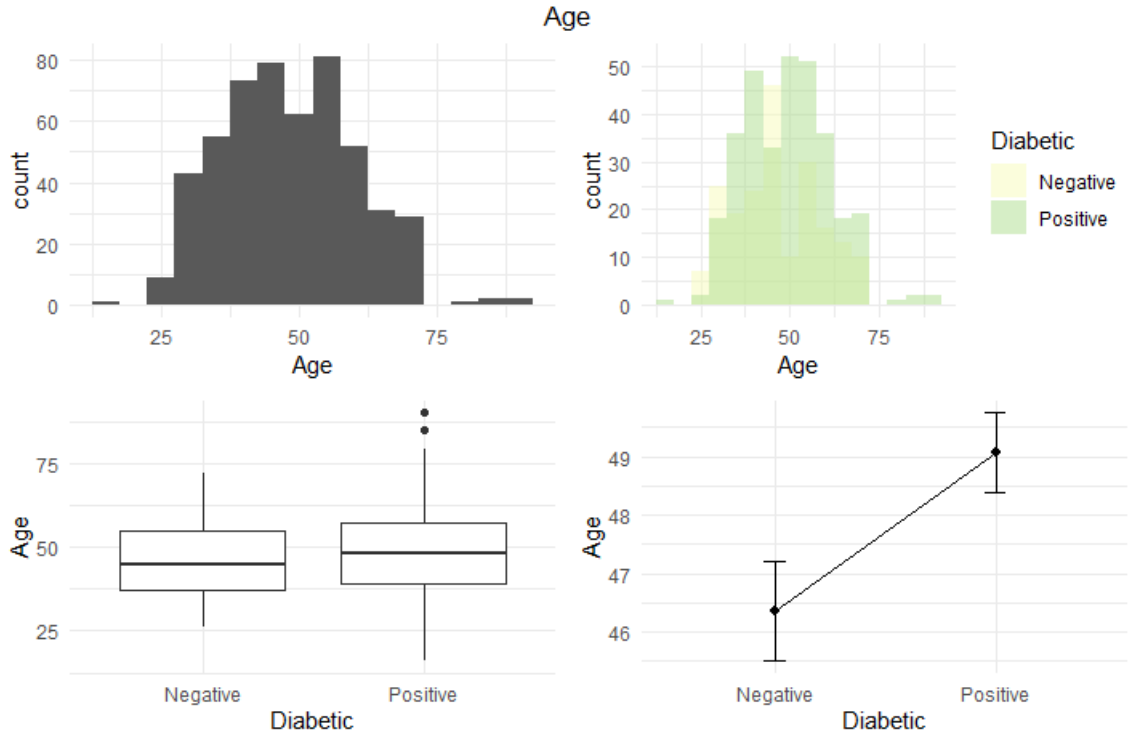
As an example, in the case of the logistic regression (LR) model, the intercept coefficient $\hat{\beta}_0$ can be adjusted to $\hat{\beta}_0^*$ according to the following transformation

$$\hat{\beta}_0^* = \hat{\beta}_0 + log\frac{\pi}{1-\pi} - log\frac{\tilde{\pi}}{1-\tilde{\pi}}$$

where $\tilde{\pi}$ is the ratio of positive outcomes in the dataset, π is the known population ratio of diabetics.

Numeric Feature

Age is the only numeric feature in this dataset. At first glance, there does not appear to be a significant difference in the age of the diabetic and healthy classes.



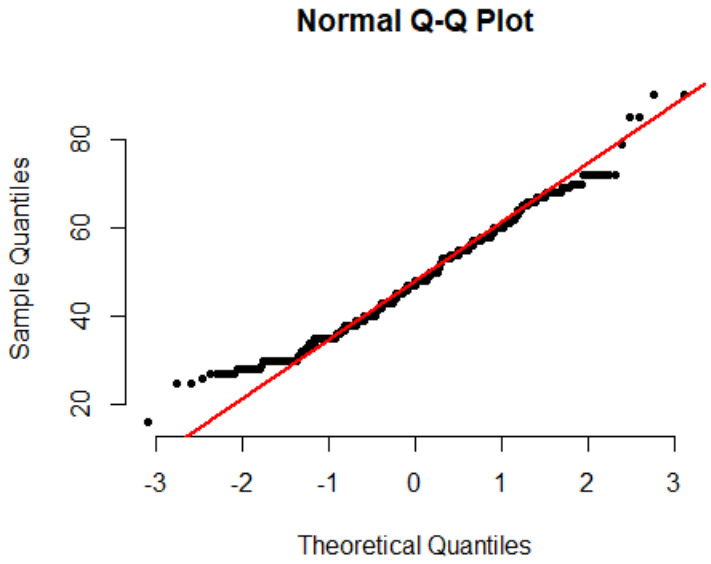
However, an ANOVA test reveals that the difference in the average ages is significant.

```
                Df Sum Sq Mean Sq F value Pr(>F)
Diabetic         1    905   905.1   6.191 0.0132 *
Residuals      518  75729   146.2
```

On average, diabetics are 2.7 years older than non-diabetics. This makes sense as diabetes prevalence is known to increase with age.

```
  Diabetic   x.Min. x.1st Qu. x.Median   x.Mean x.3rd Qu.   x.Max.
1 Negative 26.00000  37.00000 45.00000 46.36000  55.00000 72.00000
2 Positive 16.00000  39.00000 48.00000 49.07187  57.00000 90.00000
```

Although it does not pass a formal Shapiro test of normality, from the bell-shaped histogram and the QQ plot around the central quantile, it can be acceptably treated as a normally distributed feature for practical purposes.



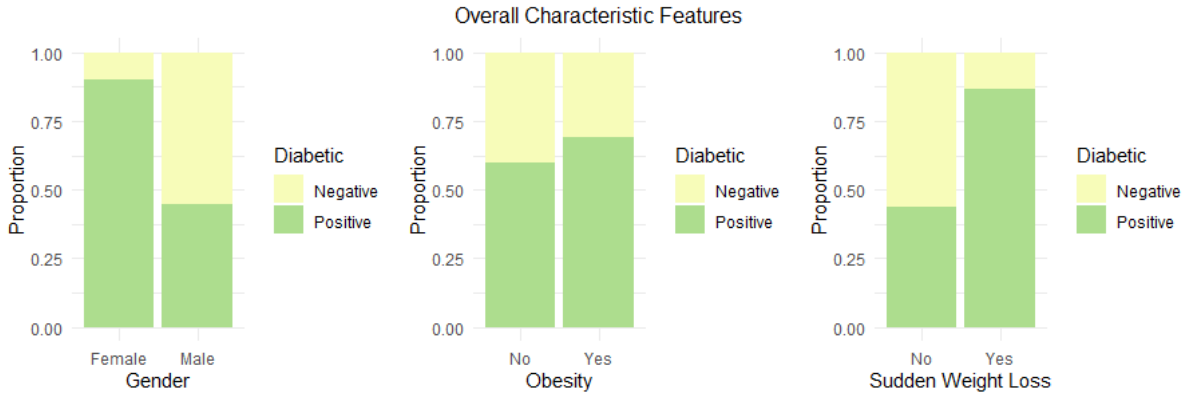
Categorical Features

All categorical features in this dataset are binary (i.e., Yes/No, Positive/Negative). Medically, they indicate the presence or absence of a symptom or illness. The simplicity of such a type of variable makes well suited for community-based detection as contrasted with more rigorous numeric measurements or multiclass responses.

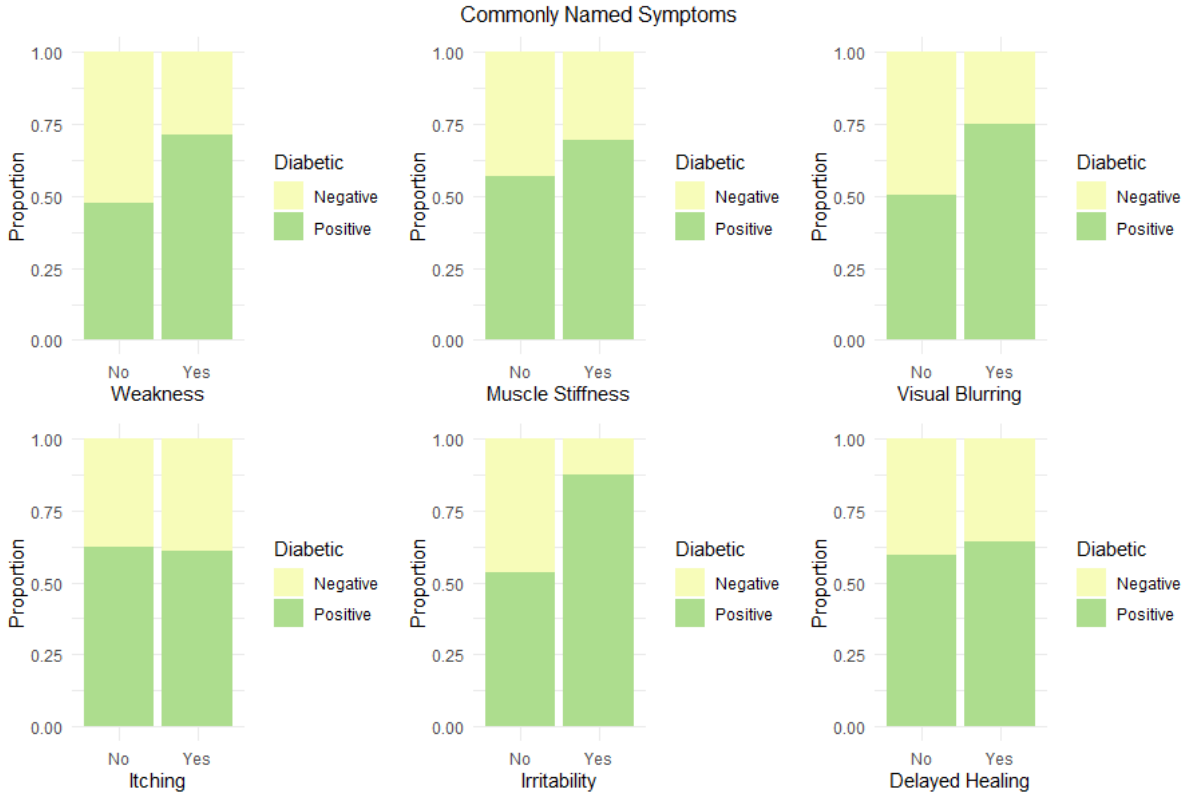
The Overall Characteristic Traits include 3 features: gender, obesity, and sudden weight loss. Females are overrepresented in the diabetes class, since globally males have a higher incidence of diabetes. Although not explored in this analysis, such a

sample bias could be corrected through assigning different weights to different genders, or through resampling techniques.

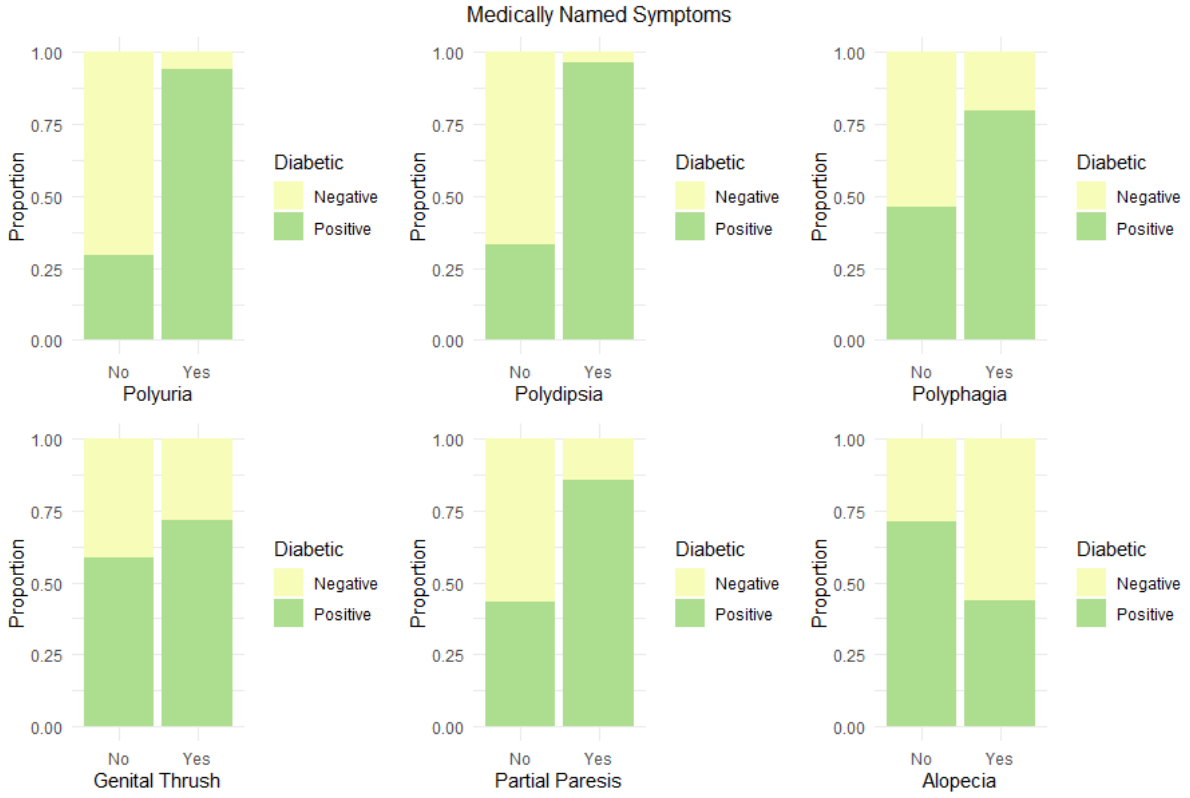
Regarding the remaining features, in line with common medical knowledge, obese people have a higher incidence of diabetes. Furthermore, sudden and unintentional weight loss is even more common in diabetics.



As for the Commonly Named Symptoms, diabetic patients are more likely to present signs of weakness, muscle stiffness, visual blurring, and irritability. Itching and delayed healing do not appear to be significantly different across the two groups.

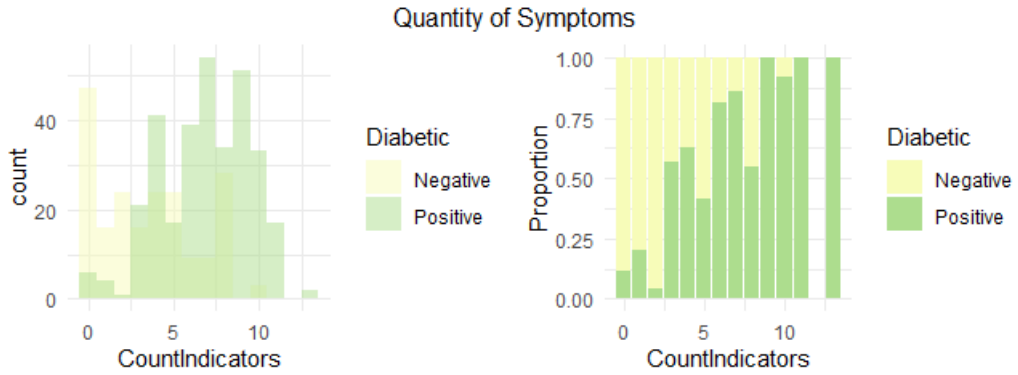


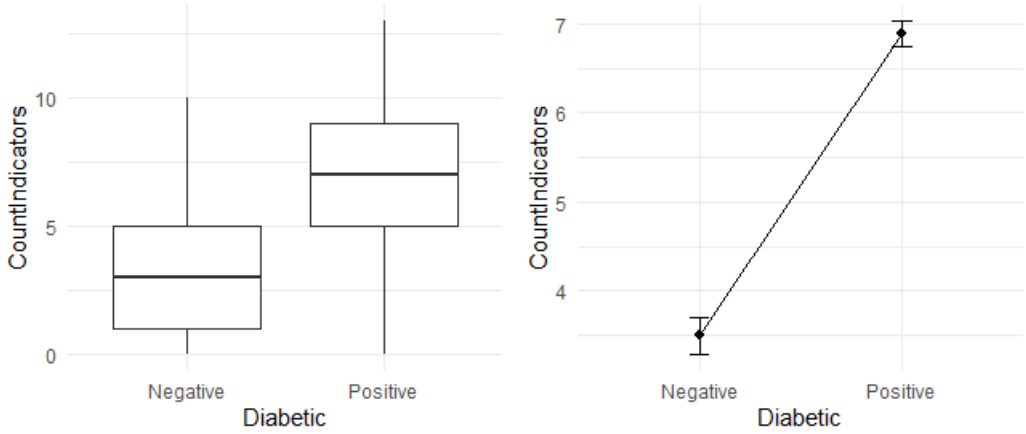
For Medically Named Symptoms, loss of hair (Alopeica) seems to be less prevalent in diabetics. Otherwise, excessive urination, thirst, or hunger, as well as partial muscular weakness or genital thrush are more prevalent in diabetics.



Feature Engineering

A new feature was created by simply counting how many symptoms each patient reported in the questionnaire. The quantity of symptoms increases the chance of a diabetic status. ANOVA reveals a significant difference in the average number of symptoms reported by the 2 classes: non-diabetics 3.5, diabetics 6.9.





Data Split

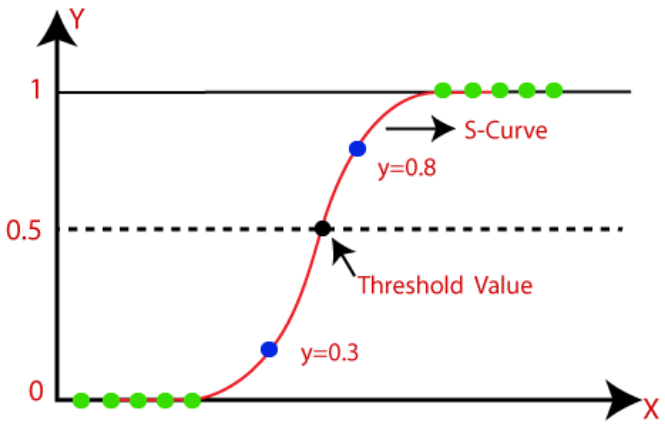
The dataset is split into a training and a testing set. The training set contains 70% of the records, with both groups maintaining the ratio of the target variable classes.

Logistic Regression

Logistic regression takes as an input multiple features and output a value between 0 and 1. This value denoted $p(X)$ is the conditional probability that the response variable is "positive" given X, denoted as $\Pr(Y = 1|X)$. It takes the form of

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_n X_n}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_n X_n}}$$

In a logistic regression, the class of the response variable is determined based on a threshold, the default being 0.5. Probabilities above the threshold are assigned the positive class, while those lower the negative, as illustrated in the following sketch.



Below is the logistic regression fit

```
Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)            2.36265    1.36587   1.730  0.08367 .
Age                   -0.06586    0.03447  -1.911  0.05606 .
GenderMale            -4.04206    0.77206  -5.235 1.65e-07 ***
PolyuriaYes            5.16819    1.24070   4.166 3.11e-05 ***
PolydipsiaYes          6.82621    1.48174   4.607 4.09e-06 ***
sudden.weight.lossYes  0.78488    0.72712   1.079  0.28039
weaknessYes            2.39592    0.75820   3.160  0.00158 **
PolyphagiaYes          0.99165    0.70983   1.397  0.16241
Genital.thrushYes      2.49791    0.80174   3.116  0.00184 **
visual.blurringYes     1.91706    0.94529   2.028  0.04256 *
ItchingYes            -3.84983    0.97118  -3.964 7.37e-05 ***
IrritabilityYes        0.66696    0.96321   0.692  0.48866
delayed.healingYes     0.62316    0.81488   0.765  0.44443
partial.paresisYes     2.37005    0.79030   2.999  0.00271 **
muscle.stiffnessYes   -1.55000    0.90375  -1.715  0.08633 .
AlopeciaYes           -0.67015    0.90816  -0.738  0.46056
ObesityYes             0.16864    0.91390   0.185  0.85360
```

The confusion matrix indicates a good performance by the logistic regression. It manages to accurately guess both classes (indicating the aforementioned imbalance in the class records is not severe so as to hamper the model's performance).

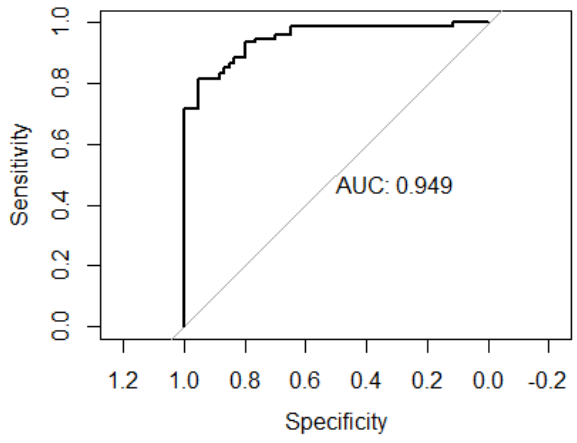
Training Dataset

```
          Reference
Prediction Negative Positive
  Negative      130       11
  Positive       10      213

               Accuracy : 0.9423
                  Kappa : 0.8783
             Sensitivity : 0.9509
             Specificity : 0.9286
      Balanced Accuracy : 0.9397
```

Test Dataset

```
          Reference
Prediction Negative Positive
  Negative       48        7
  Positive       12       89

               Accuracy : 0.8782
                  Kappa : 0.7386
             Sensitivity : 0.9271
             Specificity : 0.8000
      Balanced Accuracy : 0.8635
```

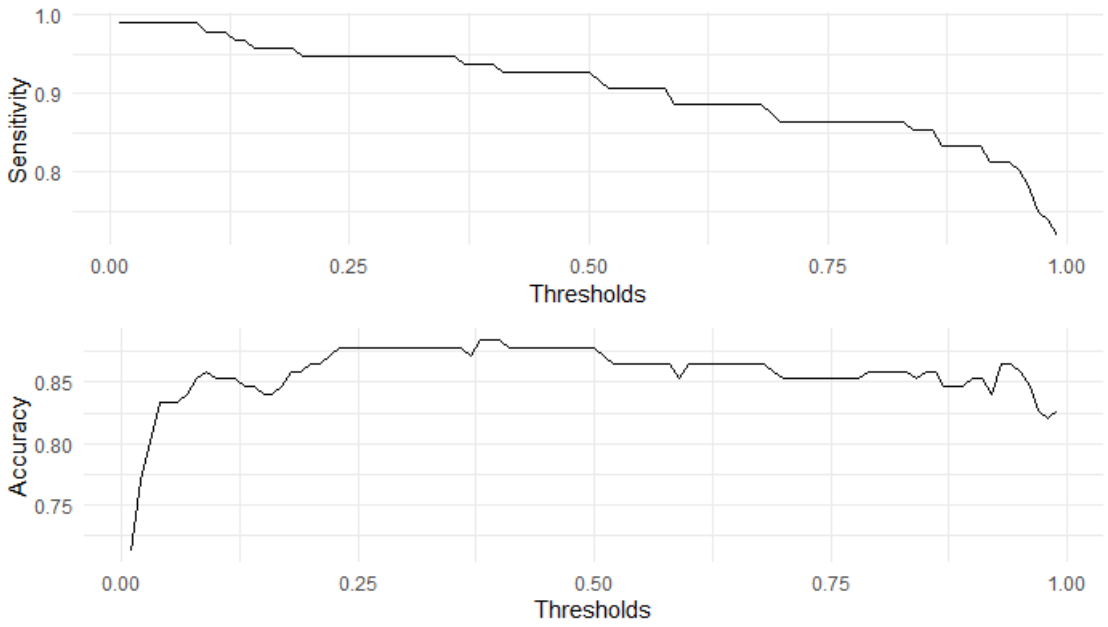
The ROC curve indicates a good performance, with 0.949 area under the curve (AUC).



In this model, the accuracy and the sensitivity are top metrics to optimize. Sensitivity is the proportion of True Positives (TP) (i.e, correctly predicted as being diabetic) to the sum of all diabetic patients. This metric is especially important in models that attempt to predict the presence of an illness. It is a worse error to predict that a patient does not have an illness and be wrong than the other way around.

$$Sensitivity = \frac{TP}{TP + FN}$$

While the default threshold of 0.5 performs admirably achieving an 87.82% test accuracy, other thresholds between 0 and 1 were explored at 0.01 increment. Based on the below plots of accuracy and sensitivity, it appears that a threshold of 0.5 is good, but not optimal.



Two options were explored:

- **Maximizing Accuracy:** a threshold between 0.38 and 0.4, inclusive, increases the test accuracy from 87.8% to 88.5%, while also improving sensitivity from 92.7% to 93.75% as a side benefit
- **Conservatively Increasing Sensitivity:** a threshold between 0.23 and 0.36, inclusive, maintains the baseline accuracy of 87.8% while increasing the sensitivity from 92.7% to 94.79%

Option 1 was selected moving forward, and the logistic regression metrics were recalculated based on a threshold of 0.4.

Training Dataset

```
          Reference
Prediction Negative Positive
  Negative      127        9
  Positive       13      215

               Accuracy : 0.9396
                  Kappa : 0.8716
             Sensitivity : 0.9598
             Specificity : 0.9071
          Balanced Accuracy : 0.9335
```

Test Dataset

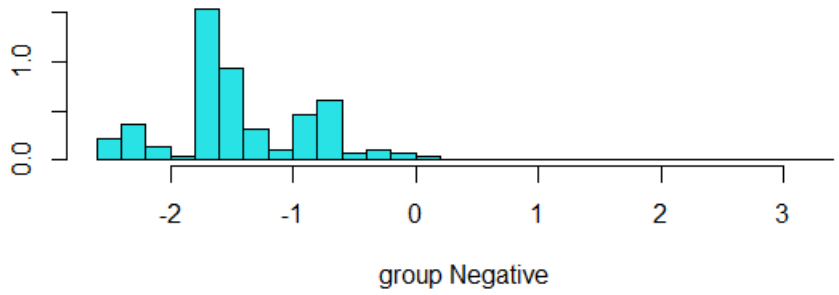
```
          Reference
Prediction Negative Positive
  Negative       48        6
  Positive       12       90

               Accuracy : 0.8846
                  Kappa : 0.7516
             Sensitivity : 0.9375
             Specificity : 0.8000
          Balanced Accuracy : 0.8688
```

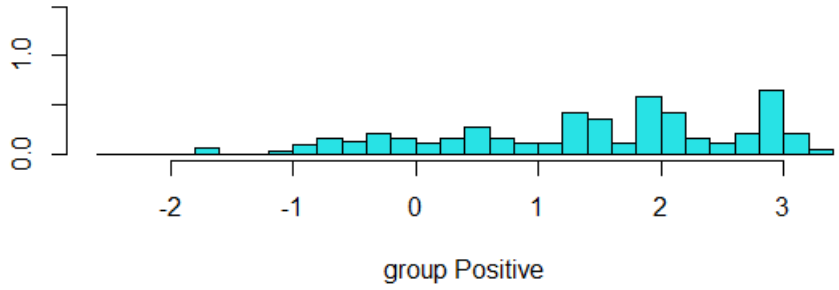
Linear Discriminant Analysis (LDA)

Fitting the training dataset using an LDA model yields the following fit:

```
Prior probabilities of groups:
 Negative  Positive
0.3846154 0.6153846

Group means:
             Age GenderMale PolyuriaYes PolydipsiaYes sudden.weight.lossYes
Negative 46.33571  0.9000000  0.06428571    0.02857143            0.1000000
Positive 49.28571  0.4464286  0.75446429    0.69642857            0.6071429
         weaknessYes PolyphagiaYes Genital.thrushYes visual.blurringYes
Negative   0.4071429     0.2714286         0.1642857          0.2857143
Positive   0.6830357     0.6250000         0.2366071          0.5580357
         ItchingYes IrritabilityYes delayed.healingYes partial.paresisYes
Negative  0.4857143      0.07857143          0.4285714          0.1571429
Positive  0.4598214      0.31696429          0.5133929          0.6071429
         muscle.stiffnessYes AlopeciaYes ObesityYes
Negative           0.2928571   0.5071429  0.1071429
Positive           0.3928571   0.2366071  0.2098214

Coefficients of linear discriminants:
                              LD1
Age                   -0.005943079
GenderMale            -0.980108335
PolyuriaYes            1.245050477
PolydipsiaYes          1.178974547
sudden.weight.lossYes  0.480002704
weaknessYes            0.139237215
PolyphagiaYes          0.095776050
Genital.thrushYes      0.729598485
visual.blurringYes     0.408403558
ItchingYes            -0.515629176
IrritabilityYes        0.689092932
delayed.healingYes    -0.193787515
partial.paresisYes     0.400781702
muscle.stiffnessYes   -0.242686073
AlopeciaYes            0.005654813
ObesityYes            -0.164883708
```

The plot below shows the spread of the linear combination of the two most dominant lags in the LDA. The two response classes have different centers and spreads, indicating that they can be distinguished well by this model.





LDA's performance is comparable to the logistic regression's.

Training Dataset

```
          Reference
Prediction Negative Positive
  Negative      138       31
  Positive        2      193

               Accuracy : 0.9093
                  Kappa : 0.8156
             Sensitivity : 0.8616
             Specificity : 0.9857
          Balanced Accuracy : 0.9237
```

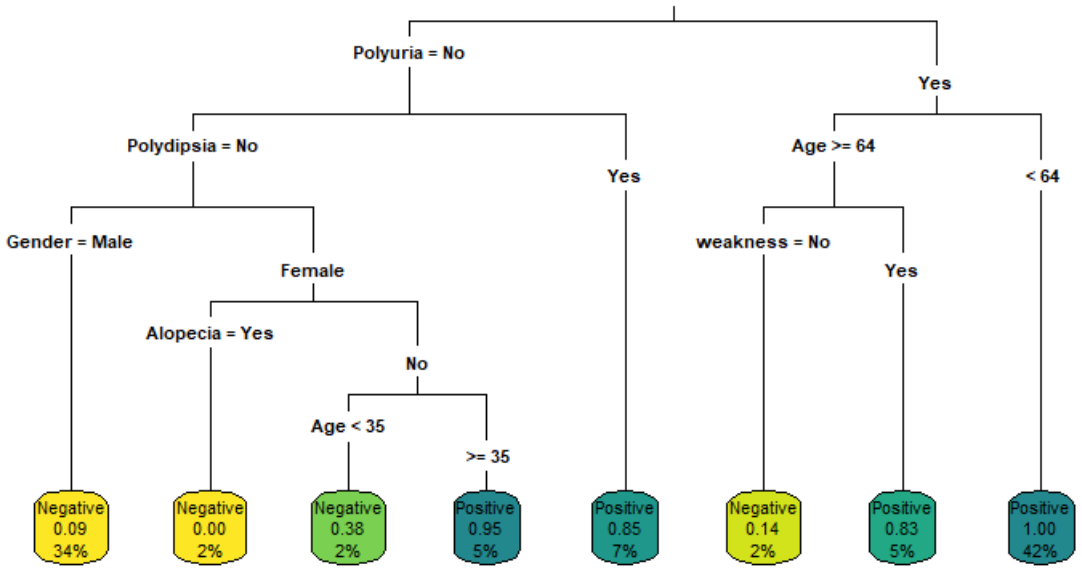
Test Dataset

```
          Reference
Prediction Negative Positive
  Negative       53       11
  Positive        7       85

               Accuracy : 0.8846
                  Kappa : 0.7593
             Sensitivity : 0.8854
             Specificity : 0.8833
          Balanced Accuracy : 0.8844
```

Decision Tree

Tree based models are simple and interpretable. This method segments the prediction space into several simple regions. At each step, the variable and threshold yielding the best separation is chosen. At the final level, the leaf assigns the majority class to the data points as a prediction.

The fitted tree model for this dataset is below. It can be easily read by starting at the root and moving along the paths of the data point until a leaf is reached. As an example, the right-most left assigns a "positive" diabetes prediction to any patient who has polyuria and is under 64 years old.



This model's performance confusion matrix and select performance metrics are below.

Training Dataset

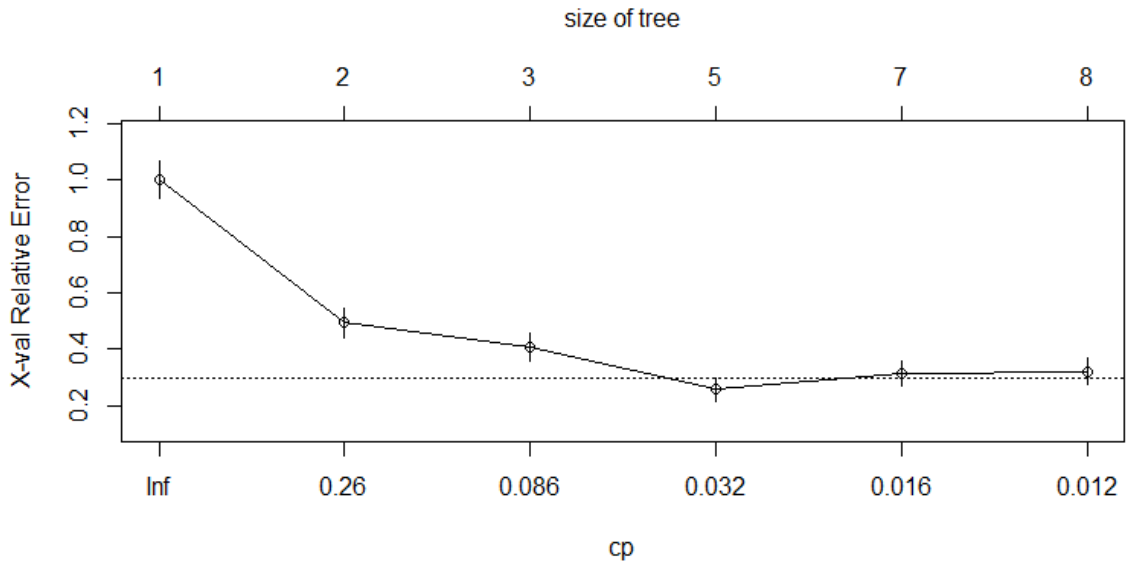
```
          Reference
Prediction Negative Positive
  Negative      132       15
  Positive        8      209

               Accuracy : 0.9368
                  Kappa : 0.8678
            Sensitivity : 0.9330
            Specificity : 0.9429
         Balanced Accuracy : 0.9379
```

Test Dataset

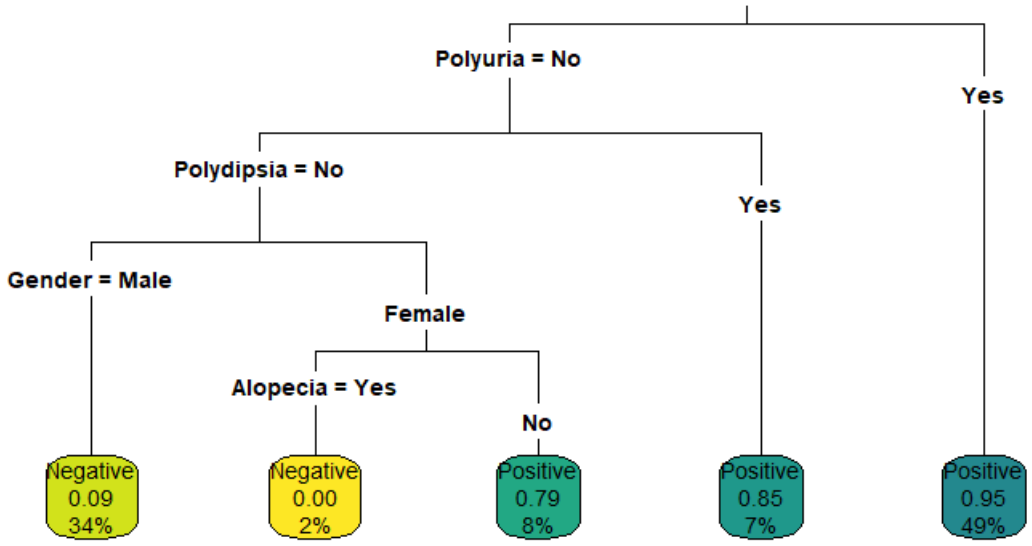
```
          Reference
Prediction Negative Positive
  Negative       52        8
  Positive        8       88

               Accuracy : 0.8974
                  Kappa : 0.7833
            Sensitivity : 0.9167
            Specificity : 0.8667
         Balanced Accuracy : 0.8917
```

Pruning the Tree

The earlier decision tree achieves a training accuracy of 93.68%, but a testing accuracy of 89.74%. This hints at a potential overfitting problem. This situation can be remedied through pruning the tree. The idea is to achieve a smaller tree with fewer splits, possibly lowering the variance, improving the interpretability, although at the cost of a little more bias.



Based on the above plot, a complexity parameter (CP) of 0.03 is chosen as an optimal threshold to prune the previously constructed decision tree.



The above pruning process actually improves the accuracy of the model. The training error is reduced, but the test error is increased. This is a typical narrowing of the overfitting gap.

Training Dataset

```
          Reference
Prediction Negative Positive
  Negative      121       11
  Positive       19      213

               Accuracy : 0.9176
                  Kappa : 0.8240
             Sensitivity : 0.9509
             Specificity : 0.8643
      Balanced Accuracy : 0.9076
```

Test Dataset

```
          Reference
Prediction Negative Positive
  Negative       50        3
  Positive       10       93

               Accuracy : 0.9167
                  Kappa : 0.8200
             Sensitivity : 0.9688
             Specificity : 0.8333
      Balanced Accuracy : 0.9010
```

Random Forest

Bagging is a general procedure to reduce the variance of a statistical learning method (and thus improve the model's performance). This is achieved by taking repeated samples from the same training dataset, building multiple trees, and taking the average of all predictions.

Random Forests improve on bagged tree by decorrelating the trees thus reducing the variance further. This is achieved by randomizing the selection of features available to the model at each tree split.

Training Dataset

```
          Reference
Prediction Negative Positive
  Negative      139        0
  Positive        1      224

               Accuracy : 0.9973
                  Kappa : 0.9942
             Sensitivity : 1.0000
             Specificity : 0.9929
      Balanced Accuracy : 0.9964
```

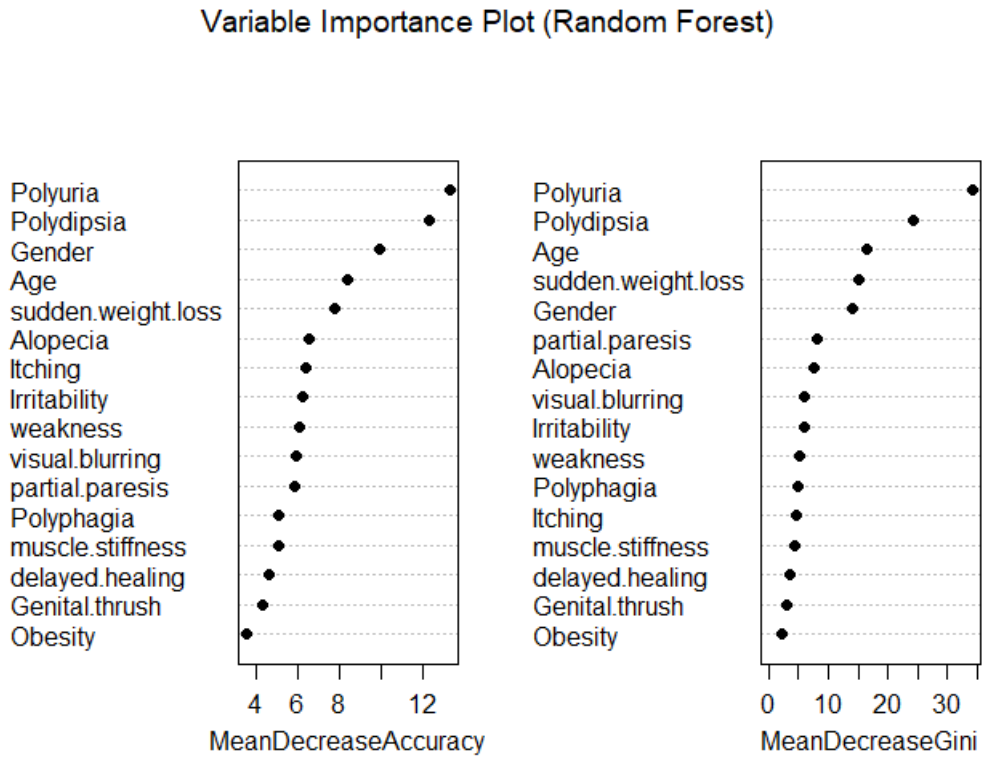
Test Dataset

```
          Reference
Prediction Negative Positive
  Negative       52        2
  Positive        8       94

               Accuracy : 0.9359
                  Kappa : 0.8620
             Sensitivity : 0.9792
             Specificity : 0.8667
      Balanced Accuracy : 0.9229
```

As expected, RF has an improved accuracy over the prior two trees.

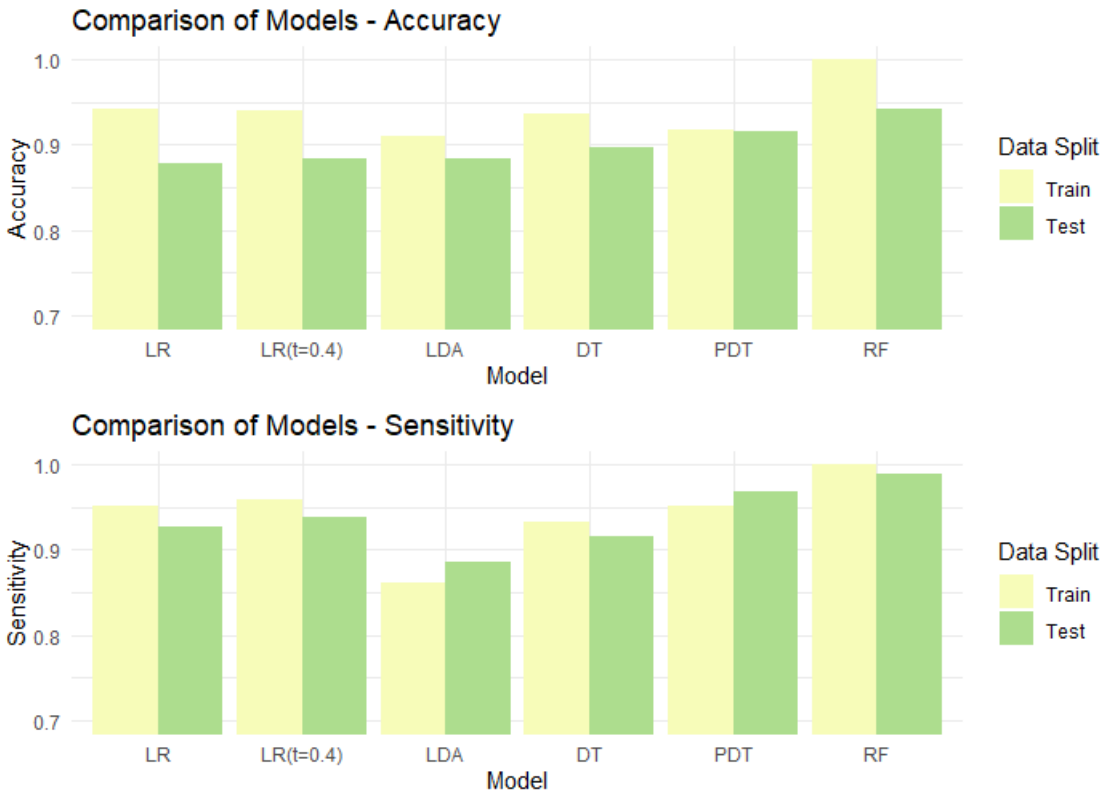
The variable importance plot computes the most important variables for the model. From both the Mean Decrease Accuracy and Mean Decrease Gini plots, it is clear that the presence of Polyuria (excessive urination), Polydipsia (excessive thirst), and

sudden weight loss are the top three non-characteristic symptoms associated with a patient's diabetes status, and hence are the most important to watch out for.



Models Comparison

- All four models achieve high accuracy and sensitivity results, confirm the validity of the concept of early detection of diabetes from the presence of certain symptoms
- The random forest (RF) model achieves both the highest accuracy and sensitivity, while the LDA has the lowest sensitivity
- The pruned decision tree (PT) is in close second place both in accuracy and sensitivity, while explaining the simplest decision model (easiest for volunteers or individuals to apply, compared to the more mathematical LR and LDA, and the digitally stored RF)
- The logistic regression (LR) provides an improved accuracy and sensitivity at the non-default threshold of 0.4



Key Findings

- Polyuria (excessive urination) and polydipsia (excessive thirst) are by far the strongest indicators of diabetes to watch out for
- Sudden weight loss is also a significant factor in most models at predicting diabete
- The higher the number of physiological symptoms present from the set studied the more likely a person is diabetic

Conclusion

Early detection of diabetes is paramount to maintain a good quality of life for patients. Lack of costly home glucose monitors or a strong healthcare infrastructure in less developed countries does not need to hinder the early detection of this disease. As shown in this study, minimal training of community volunteers, the community, and even individuals to simply keep an eye out for the presence of a few symptoms provides a strong and free predictor to identify individuals that might have diabetes at an early stage. The main symptoms to keep an eye for are excessive urination (volume or frequency), excessive thirst, and a sudden and unintentional weight loss.

Appendix: R Code

```
################################################################################
# PROJECT DETAILS


#-------------------------------------------------------------------------------
# ADMINSTRATIVE

# Name:         Shihab Hamati
# Matricola:    985941

# Module:       Statistical Learning
# Exam Date:    03 Nov 2022

# Part 1:       Supervised Learning


#-------------------------------------------------------------------------------
# REFERENCES

# Dataset:      "Early stage diabetes risk prediction dataset"
# Dataset date: 12 Jul 2020
# Link:
https://archive.ics.uci.edu/ml/datasets/Early+stage+diabetes+risk+prediction+datase
t.#

# Description:
# - A dataset consisting of questionnaire reponses from 520 patients, approved by a
doctor
# - Indicators collected are all of a physiological nature


#-------------------------------------------------------------------------------
# PERSONAL MOTIVATION

# - Diabetes is a prevalent disease in my country and my extended family

# - Models would allow non-doctors (including other medical personnel as well as
family and friends of concerned patient)
#   to look out for the most important physiological diabetes red flags
#   to seek prompt medical evaluation rather than let early stage diabetes go
#   undetected (which causes damage and becomes harder to control at later stages)

# - This is especially helpful in underdeveloped area where access to home devices
#   is not common or easy



################################################################################
# LIBRARIES

library(dplyr)
library(DataExplorer)
library(ggplot2)
library(gridExtra)
library(matrixStats)
library(ggpubr)
library(caret)
library(pROC)
library(MASS)
library(rpart)
library(rpart.plot)
library(randomForest)
library(reshape2)
library(ggbreak)
```

```
################################################################################
# DATA SETUP

# Download dataset directly from online source
data_url= "https://archive.ics.uci.edu/ml/machine-learning-
databases/00529/diabetes_data_upload.csv"
data <- read.csv(data_url, header=TRUE, stringsAsFactors = TRUE)

# Option to read data from user selected local destination
#data <- read.csv(choose.files(), header=TRUE, stringsAsFactors = TRUE)

# view summary of data
summary(data)
str(data)
head(data)

colnames(data)
colnames(data)[17] <- "Diabetic"

# check for missing data
sum(is.na(data))

# quickly create report to explore data
# create_report(data)


#-------------------------------------------------------------------------------
# DESCRIPTION
# - Polyuria    : excessive urination (frequency or volume)
# - Polydipsia  : excessive thirst
# - Polyphagia  : excessive eating
# - Paresis     : muscular weakness (partial)
# - Alopecia    : bodily hair loss



################################################################################
# EXPLORATORY DATA ANALYSIS (EDA)


#-------------------------------------------------------------------------------
# RESPONSE VARIABLE
p0 <- ggplot(data, aes(x = Diabetic)) +
  geom_bar(aes(fill = Diabetic)) +
  geom_text(aes(y = ..count..,
                label = paste0(round(prop.table(..count..),3) * 100, '%')),
            stat = 'count') +
  ggtitle("Distribution of Response Variable in Dataset") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")
p0


#-------------------------------------------------------------------------------
# NUMERIC FEATURE
p1a <- ggplot(data = data, aes(x = Age)) +
  geom_histogram(binwidth = 5) +
  theme_minimal()

p1b <- ggplot(data = data, aes(x = Age, group = Diabetic, fill = Diabetic)) +
  geom_histogram(position = "identity", alpha = 0.5, binwidth = 5) +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

p1c <- ggplot(data = data, aes(x = Diabetic, y = Age)) +
  geom_boxplot() +
  theme_minimal()
```

```
# Statistical summary of Age, grouped by Diabetes status
with(data, aggregate(Age, list(Diabetic = Diabetic), FUN = summary))

p1d <- ggline(data, x = "Diabetic", y = "Age", add = "mean_se") +
  theme_minimal()

# Anova
aov_age <- aov(Age ~ Diabetic, data = data)
summary(aov_age) # p-value < 0.001 indicating significant difference

# Display Plots
grid.arrange(p1a, p1b, p1c, p1d, ncol=2, top = "Age")

# Explore normality
qqnorm(data$Age, pch = 20, frame = FALSE)
qqline(data$Age, col="red", lwd = 2)


#-------------------------------------------------------------------------
# CATEGORICAL FEATURES

#.........................................................................
# Overall characteristics

p2 <- ggplot(data, aes(x = Gender, fill = Diabetic)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

p3 <- ggplot(data, aes(x = Obesity, fill = Diabetic)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

p4 <- ggplot(data, aes(x = sudden.weight.loss, fill = Diabetic)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  xlab("Sudden Weight Loss") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

#.........................................................................
# Commonly named symptoms

p5 <- ggplot(data, aes(x = weakness, fill = Diabetic)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  xlab("Weakness") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

p6 <- ggplot(data, aes(x = muscle.stiffness, fill = Diabetic)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  xlab("Muscle Stiffness") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

p7 <- ggplot(data, aes(x = visual.blurring, fill = Diabetic)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  xlab("Visual Blurring") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")
```

```
p8 <- ggplot(data, aes(x = Itching, fill = Diabetic)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

p9 <- ggplot(data, aes(x = Irritability, fill = Diabetic)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

p10 <- ggplot(data, aes(x = delayed.healing, fill = Diabetic)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  xlab("Delayed Healing") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

#.....................................................................
# Medically named symptoms

p11 <- ggplot(data, aes(x = Polyuria, fill = Diabetic)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

p12 <- ggplot(data, aes(x = Polydipsia, fill = Diabetic)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

p13 <- ggplot(data, aes(x = Polyphagia, fill = Diabetic)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

p14 <- ggplot(data, aes(x = Genital.thrush, fill = Diabetic)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  xlab("Genital Thrush") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

p15 <- ggplot(data, aes(x = partial.paresis, fill = Diabetic)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  xlab("Partial Paresis") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

p16 <- ggplot(data, aes(x = Alopecia, fill = Diabetic)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

# Display Plots

grid.arrange(p2, p3, p4,
             ncol=3, top = "Overall Characteristic Traits")

grid.arrange(p5, p6, p7, p8, p9, p10,
             ncol=3, top = "Commonly Named Symptoms")
```

```
grid.arrange(p11, p12, p13, p14, p15, p16,
             ncol=3, top = "Medically Named Symptoms")


#-------------------------------------------------------------------------------
# FEATURE ENGINEERING

# Explore relation of how many indicators exist with outcome
# create new column counting indicators for each row
CountIndicators <-
  rowCounts(as.matrix(data), cols = colnames(data)[3:16], value = "Yes")

data_ci <- data.frame(CountIndicators, data["Diabetic"]) # separate df

# Statistical summary of Count of Indicators, grouped by Diabetes status
with(data_ci, aggregate(CountIndicators,
                        list(Diabetic = Diabetic), FUN = summary))

# Plots
p17a <- ggplot(data = data_ci,
               aes(x = CountIndicators, group = Diabetic, fill = Diabetic)) +
  geom_histogram(position = "identity", alpha = 0.5, binwidth = 1) +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

p17b <- ggplot(data, aes(x = CountIndicators, fill = Diabetic)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

p17c <- ggplot(data = data_ci, aes(x = Diabetic, y = CountIndicators)) +
  geom_boxplot()  +
  theme_minimal()

p17d <- ggline(data_ci,
               x = "Diabetic", y = "CountIndicators", add = "mean_se") +
  theme_minimal()

grid.arrange(p17a, p17b, p17c, p17d, ncol=2, top = "Quantity of Symptoms")

# Statistical values appears different for Diabetics vs Non-Diabetic
# Test if the mean count of indicators are statistically different

# Anova
aov_ci <- aov(CountIndicators ~ Diabetic, data = data)
summary(aov_ci) # p-value < 0.001 indicating significant difference

# This feature will not be provided to the models, so it was kept in its own df
# All models are able to account for it in some way, and it is of interest
# to explore the explanatory power of each physiological feature,
# especially in importance plots in RF



#################################################################################
# MODELS

# Splitting Datasets
set.seed(1234)
split_train_test <- createDataPartition(data$Diabetic,p=0.7,list=FALSE)
dtrain<- data[split_train_test,]
dtest<- data[-split_train_test,]
```

```
#-------------------------------------------------------------------------------
# LOGISTIC REGRESSION

lr_fit <- glm(Diabetic ~ ., data=dtrain, family=binomial(link='logit'))
summary(lr_fit)

# Confusion Matrices and Accuracies for LR
# Train set
lr_prob_dtrain <- predict(lr_fit, dtrain, type="response")
lr_pred_dtrain <- ifelse(lr_prob_dtrain > 0.5, "Positive", "Negative")
table(Predicted = lr_pred_dtrain, Actual = dtrain$Diabetic)
mean(lr_pred_dtrain == dtrain$Diabetic)

# confirm with built-in function
cm_lr_dtrain <- confusionMatrix(
  as.factor(lr_pred_dtrain),
  as.factor(dtrain$Diabetic),
  positive = "Positive"
)
cm_lr_dtrain

# Test set
lr_prob_dtest <- predict(lr_fit, dtest, type="response")
lr_pred_dtest <- ifelse(lr_prob_dtest > 0.5, "Positive", "Negative")
table(Predicted = lr_pred_dtest, Actual = dtest$Diabetic)
mean(lr_pred_dtest == dtest$Diabetic)

# confirm with built-in function
cm_lr_dtest <- confusionMatrix(
  as.factor(lr_pred_dtest),
  as.factor(dtest$Diabetic),
  positive = "Positive"
)
cm_lr_dtest

# ROC Curve
test_roc = roc(dtest$Diabetic ~ lr_prob_dtest, plot = TRUE, print.auc = TRUE)
as.numeric(test_roc$auc)

# Explore threshold
lr_thresholds <- c()
lr_sensitivities <- c()
lr_accuracies <- c()

for(t in 1:99){
  lr_pred_t <- ifelse(lr_prob_dtest > t/100.0, "Positive", "Negative")
  cm_t <- table(Predicted = lr_pred_t, Actual = dtest$Diabetic)
  lr_thresholds <- append(lr_thresholds, t/100.0)
  lr_sensitivities <- append(lr_sensitivities,
                             sensitivity(cm_t, positive = "Positive"))
  lr_accuracies <- append(lr_accuracies, mean(lr_pred_t == dtest$Diabetic))
}

# Plot changes in Sensitivity (correct positives) and Accuracy
p18 <- ggplot(data=data.frame(lr_thresholds, lr_sensitivities),
              aes(x = lr_thresholds, y = lr_sensitivities)) +
  geom_line() +
  labs(x = 'Thresholds',y='Sensitivity') +
  theme_minimal()

p19 <- ggplot(data=data.frame(lr_thresholds, lr_accuracies),
              aes(x = lr_thresholds, y = lr_accuracies)) +
  geom_line() +
  labs(x = 'Thresholds',y='Accuracy') +
  theme_minimal()

grid.arrange(p18, p19, ncol=1)
```

```
# Option 1: Optimization of Sensitivity
max(lr_sensitivities)
lr_thresholds[which(lr_sensitivities == max(lr_sensitivities))]
lr_accuracies[which(lr_sensitivities == max(lr_sensitivities))]

# the optimum sensitivity is achieved at t <= 0.09
# the best accuracy achievable in this range is 85.9% at t = 0.09

lr_sensitivities[23:36]
lr_accuracies[23:36]

# a good balance between both metrics could be 0.23<= t <=0.36
# it increases sensitivity (from 92.7% to 94.79%)
# without lowering accuracy at all (from default 87.8% at t = 0.5)
# best sensitivity is achieved at low thresholds but accuracy plunges alot

# Option 2: Optimization of Accuracy
max(lr_accuracies)
lr_thresholds[which(lr_accuracies == max(lr_accuracies))]
lr_sensitivities[which(lr_accuracies == max(lr_accuracies))]

# another good point is 0.38 <= t <= 0.4
# it increases test accuracy from default t = 0.5 (from 87.8% to 88.5%)
# while also increasing test sensitivity (from 92.7% to 93.75%) - lucky bonus

# Conclusion of LR Threshold
# The default t = 0.5 is good, but not optimum in either scenarios, hence:
# To optimize sensitivity (and luckily without loss of acc): t = 0.23-0.36
# To optimize accuracy (and luckily sensitivity in this case): t = 0.38-0.40

# Re-fit Logistic Regression
# using option 2, since acc is the metric used to compare the different models

t = 0.4

lr_pred_dtrain_t <- ifelse(lr_prob_dtrain > t, "Positive", "Negative")
lr_pred_dtest_t <- ifelse(lr_prob_dtest > t, "Positive", "Negative")

cm_lr_dtrain_t <- confusionMatrix(
  as.factor(lr_pred_dtrain_t),
  as.factor(dtrain$Diabetic),
  positive = "Positive"
)
cm_lr_dtrain_t

cm_lr_dtest_t <- confusionMatrix(
  as.factor(lr_pred_dtest_t),
  as.factor(dtest$Diabetic),
  positive = "Positive"
)
cm_lr_dtest_t


#-------------------------------------------------------------------------------
# LINEAR DISCRIMINANT ANALYSIS (LDA)

lda_fit = lda(Diabetic ~ ., data=dtrain)
lda_fit
plot(lda_fit)

# Confusion Matrices and Accuracies of LDA

# Training dataset
lda_pred_dtrain = predict(lda_fit, dtrain)$class
table(lda_pred_dtrain, dtrain$Diabetic)
mean(lda_pred_dtrain == dtrain$Diabetic)
```

```
# confirm with built-in function
cm_lda_dtrain <- confusionMatrix(
  as.factor(lda_pred_dtrain),
  as.factor(dtrain$Diabetic),
  positive = "Positive"
)
cm_lda_dtrain

# Test dataset
lda_pred_dtest = predict(lda_fit, dtest)$class
table(lda_pred_dtest, dtest$Diabetic)
mean(lda_pred_dtest == dtest$Diabetic)

# confirm with built-in function
cm_lda_dtest <-confusionMatrix(
  as.factor(lda_pred_dtest),
  as.factor(dtest$Diabetic),
  positive = "Positive"
)
cm_lda_dtest


#------------------------------------------------------------------------------
# DECISION TREE
tree <- rpart(formula = Diabetic ~ ., data=dtrain)
printcp(tree)
rpart.plot(tree, type=3, box.palette="YlGn")

tree_pred_dtrain = predict(tree, dtrain, type="class")
tree_pred_dtest = predict(tree, dtest, type="class")

cm_tree_dtrain <- confusionMatrix(
  as.factor(tree_pred_dtrain),
  as.factor(dtrain$Diabetic),
  positive = "Positive"
)
cm_tree_dtrain

cm_tree_dtest <- confusionMatrix(
  as.factor(tree_pred_dtest),
  as.factor(dtest$Diabetic),
  positive = "Positive"
)
cm_tree_dtest

plotcp(tree) # to choose cp corresponding to lowest X-val relative error


#------------------------------------------------------------------------------
# PRUNED DECISION TREE

ptree <- prune(tree, cp = 0.03)
printcp(ptree)
rpart.plot(ptree, type=3, box.palette="YlGn")

ptree_pred_dtrain = predict(ptree, dtrain, type="class")
ptree_pred_dtest = predict(ptree, dtest, type="class")

cm_ptree_dtrain <- confusionMatrix(
  as.factor(ptree_pred_dtrain),
  as.factor(dtrain$Diabetic),
  positive = "Positive"
)
cm_ptree_dtrain

cm_ptree_dtest <- confusionMatrix(
  as.factor(ptree_pred_dtest),
  as.factor(dtest$Diabetic),
```

```
  positive = "Positive"
)
cm_ptree_dtest


#-------------------------------------------------------------------------------
# RANDOM FOREST

rf = randomForest(Diabetic ~ ., data = dtrain,
                  ntree = 50, mtry = 3, importance = TRUE)

varImpPlot(rf, bg = "black",
           main = "Variable Importance Plot (Random Forest)")

rf_pred_dtrain <- predict(rf, dtrain)
rf_pred_dtest <- predict(rf, dtest)

cm_rf_dtrain <- confusionMatrix(
  as.factor(rf_pred_dtrain),
  as.factor(dtrain$Diabetic),
  positive = "Positive"
)
cm_rf_dtrain

cm_rf_dtest <- confusionMatrix(
  as.factor(rf_pred_dtest),
  as.factor(dtest$Diabetic),
  positive = "Positive"
)
cm_rf_dtest

# RF models are not prone to overfitting
# So, the larger gap between train acc (100%) and test acc (94.23%)
# does not indicate a potential to improve test acc (like in other models)

# Also, RF achieves the highest train and test accuracies anyway

plot(dtrain$Diabetic, rf_pred_dtrain)
plot(dtest$Diabetic, rf_pred_dtest)



################################################################################
# SUMMARY

abbr <- c("1a. LR", "1b. LR(t=0.4)", "2. LDA", "3a. DT", "3b. PDT", "4. RF")

fullname <- c("Logistic Regression",
              "Logistic Regression(thresh=0.4)",
              "Linear Discrimnant Analysis",
              "Decision Tree",
              "Pruned Decision Tree",
              "Random Forest")

# Retrieve values from stored confusion matrices for accuracy and sensitivity
# for both the training and test datasets

acc_train <- c(cm_lr_dtrain$overall["Accuracy"],
               cm_lr_dtrain_t$overall["Accuracy"],
               cm_lda_dtrain$overall["Accuracy"],
               cm_tree_dtrain$overall["Accuracy"],
               cm_ptree_dtrain$overall["Accuracy"],
               cm_rf_dtrain$overall["Accuracy"])
```

```
acc_test <- c(cm_lr_dtest$overall["Accuracy"],
              cm_lr_dtest_t$overall["Accuracy"],
              cm_lda_dtest$overall["Accuracy"],
              cm_tree_dtest$overall["Accuracy"],
              cm_ptree_dtest$overall["Accuracy"],
              cm_rf_dtest$overall["Accuracy"])

snsv_train <- c(cm_lr_dtrain$byClass["Sensitivity"],
                cm_lr_dtrain_t$byClass["Sensitivity"],
                cm_lda_dtrain$byClass["Sensitivity"],
                cm_tree_dtrain$byClass["Sensitivity"],
                cm_ptree_dtrain$byClass["Sensitivity"],
                cm_rf_dtrain$byClass["Sensitivity"])

snsv_test <- c(cm_lr_dtest$byClass["Sensitivity"],
               cm_lr_dtest_t$byClass["Sensitivity"],
               cm_lda_dtest$byClass["Sensitivity"],
               cm_tree_dtest$byClass["Sensitivity"],
               cm_ptree_dtest$byClass["Sensitivity"],
               cm_rf_dtest$byClass["Sensitivity"])


# Manipulate dataframes for plotting purposes
acc_summary <- data.frame(abbr, fullname, acc_train, acc_test)
colnames(acc_summary)[3:4] <- c("Train", "Test")
acc_summary <- melt(acc_summary)

snsv_summary <- data.frame(abbr, fullname, snsv_train, snsv_test)
colnames(snsv_summary)[3:4] <- c("Train", "Test")
snsv_summary <- melt(snsv_summary)


#------------------------------------------------------------------------------
# MODELS COMPARISON

abbr_ticks <- c("LR", "LR(t=0.4)", "LDA", "DT", "PDT", "RF")

p20 <- ggplot(data = acc_summary, aes(x = abbr, y = value, fill = variable)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  coord_cartesian(ylim = c(.7, 1)) +
  labs(title = "Comparison of Models - Accuracy", x = "Model", y = "Accuracy",
       fill = "Data Split") +
  scale_x_discrete(labels = abbr_ticks) +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

p21 <- ggplot(data = snsv_summary, aes(x = abbr, y = value, fill = variable)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  coord_cartesian(ylim = c(.7, 1)) +
  labs(title = "Comparison of Models - Sensitivity", x = "Model" , y =
"Sensitivity",
       fill = "Data Split") +
  scale_x_discrete(labels = abbr_ticks) +
  theme_minimal() +
  scale_fill_brewer(palette = "YlGn")

grid.arrange(p20, p21, ncol=1)



##############################################################################
# END
```