Predictive Analysis in Diabetes

Cheng Yin Chuang

Statistics B.S., School of Liberal Arts, University of Minnesota

STAT4052: Statistical Machine Learning II

Dr. Sara Algeri

December 11, 2023

1. Introduction

This study aims to identify the most influential variables that contribute the most to the likelihood of diabetes and find an optimal predictive model to help predict the outcome of future patients. To achieve this, we have constructed a variety of statistical methods to classify and predict whether an individual has diabetes, utilizing a dataset with key variables such as age, gender, hypertension, heart disease history, smoking history, BMI, HbA1c level, and blood glucose level. One of the main aspects is to explore different strategies for handling missing data. This includes employing mean imputation and iterative regression imputation techniques to the data.

Using different statistical methods to predict whether an individual has diabetes or not and find out which variable among all the variables is the most important, using statistical methods include Logistic regression, LDA, QDA, KNN, and Random Forest. Overall, we found that the HbA1c level is the most important factor among all the variables.

2. Method and Materials

In this analysis, I have used various statistical methods. The first method is variable selection, this purpose is to select the most significant variables from the dataset, by adding or removing variables based on the AIC, and choosing the model with the lowest AIC.

Logistic regression is suitable for binary classification problems. It estimates the probability of the different outcomes, specifically the likelihood of one over the other.

Linear Discriminant Analysis(LDA) predicts posterior probabilities under two assumptions for this method, normally distributed predictors and equal covariance matrices across classes. LDA is an effective model to use when the dataset is small. Quadratic Discriminant Analysis(QDA). Similar to LDA, the second assumption does not hold, it allows different covariance matrices for each class. This flexibility makes QDA more applicable in situations where the second assumption in LDA does not hold.

K-Nearest Neighbors(KNN) is useful for both classification and regression cases. K is the nearest neighbor to consider. The choice of K affects the decision boundary significantly. Smaller K can lead to more complex boundaries, and could potentially lead to overfitting. While larger K tends towards a smoother boundary, might lead to underfitting.

Random Forest(RF), aims at variance reduction and focuses on the number of predictors we consider when we do the splitting and the number of trees, more trees increase the performance, with different considerations for classification and regression response variables.

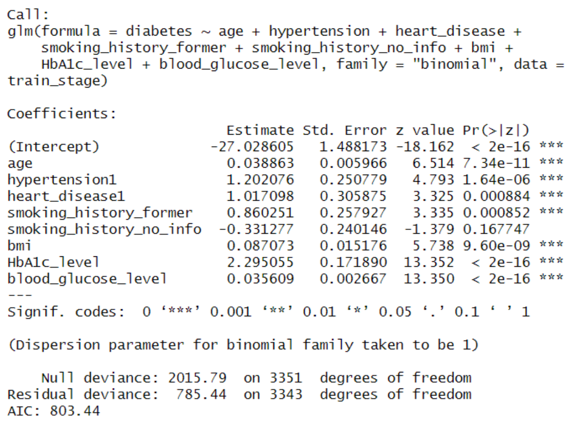
Last but not least, the Iterative Regression Imputation method handles missing data, using regression models to iteratively estimate missing values. Initially filled missing values with estimates by mean, median, or mode, and then refined estimates based on model predictions.

3. Results

3.1 Stage One

In this section, I will analyze the results, focusing on comparing the test error rates and ROC curves of various models. The initial model we explore is the logistic regression model. In this stage, termed “stage one”, we replace the missing value in the continuous variable with the mean of the variable and exclude the observations with missing value in the categorical variable, additionally, we exclude the variable “Other” in gender because there is only one observation left and might affect the model. The amount of the observations is reduced from 5000 to only 4190. We then split the data into a train set for 80 percent of the data and 20 percent of the data as a test set. We then use variable selection to find the most important variables among all the predictive variables, which are age, hypertension, heart disease, smoking history former, smoking history no info, BMI, HbA1c level, and blood glucose levels. The chosen variables will be used in stage one to fit into different models.

3.1.1 Logistic Regression Prediction

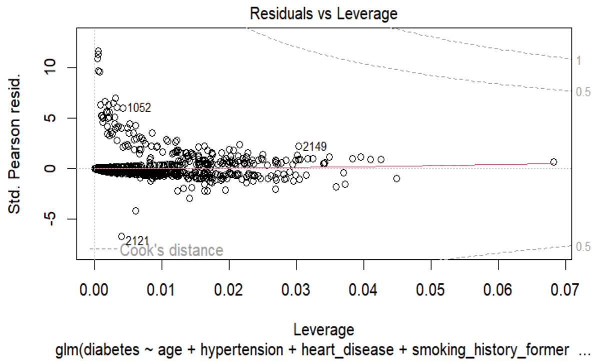


```
Call:
glm(formula = diabetes ~ age + hypertension + heart_disease +
    smoking_history_former + smoking_history_no_info + bmi +
    HbA1c_level + blood_glucose_level, family = "binomial", data =
train_stage)

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)            -27.028605   1.488173 -18.162  < 2e-16 ***
age                      0.038863   0.005966   6.514 7.34e-11 ***
hypertension1            1.202076   0.250779   4.793 1.64e-06 ***
heart_disease1           1.017098   0.305875   3.325 0.000884 ***
smoking_history_former   0.860251   0.257927   3.335 0.000852 ***
smoking_history_no_info -0.331277   0.240146  -1.379 0.167747
bmi                      0.087073   0.015176   5.738 9.60e-09 ***
HbA1c_level              2.295055   0.171890  13.352  < 2e-16 ***
blood_glucose_level      0.035609   0.002667  13.350  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2015.79  on 3351  degrees of freedom
Residual deviance:  785.44  on 3343  degrees of freedom
AIC: 803.44
```



The result of the logistic model is the image on the left, we can see that except for the variable smoking_history_no_info, all other variables were statistically significant to the response variable. The image on the right shows the diagnostic plot of the model, which is favorable, although there are some high-leverage observations, more importantly, there is no influential observation that could skew the result. We then test the Deviance and Pearson chi-square test for the model, and both p-values are larger than 0.05, meaning the model fits the observations well. Lastly, we fit the test set into the model we built for the logistic regression and we get 0.05 for the test error rate.

3.1.2 Random Forest and Bagging and Boosting

The last two models we will fit in stage one are the Random Forest and Bagging. For Random Forest, because there are only 7 predictive variables, we choose the square root of 7 as the number of predictors randomly sampled each time and the number of trees is default setting 500. For Bagging, we consider all 7 predictive variables rather than randomly sampling a few of them.

```
Call:
 randomForest(formula = diabetes ~ age + hypertension + heart_disease + 
smoking_history_former + smoking_history_no_info + bmi +      HbA1c_level + 
blood_glucose_level, data = stage1_data, mtry = 2,      importance = TRUE) 
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 2.82%
Confusion matrix:
     0   1 class.error
0 3826   0   0.0000000
1  118 246   0.3241758
```

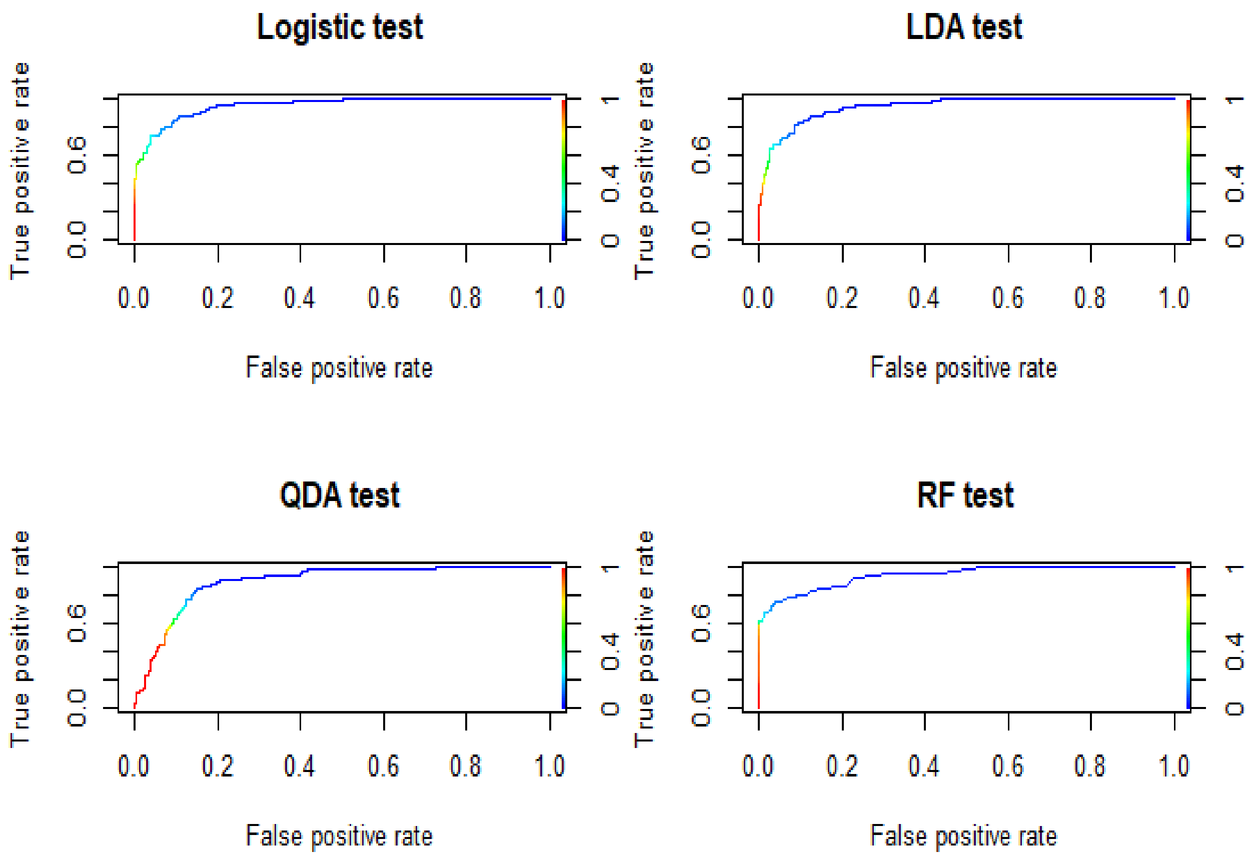
The table above is the result of Random Forest, we can see that there is an OOB estimate of error rate, which represents how well the model performed, the smaller the value of this means a better model was performed. The OOB estimate of error rate is 2.82% for Random Forest, and the table below is the result of Bagging and its OOB estimate of error rate is 3.1%. Therefore, the Random Forest is slightly better than Bagging. The error rate for the testing set is 0.031.

```
Call:
 randomForest(formula = diabetes ~ age + hypertension + heart_disease + 
smoking_history_former + smoking_history_no_info + bmi +      HbA1c_level + 
blood_glucose_level, data = stage1_data, mtry = 7,      importance = TRUE) 
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 7

        OOB estimate of  error rate: 3.1%
Confusion matrix:
     0   1 class.error
0 3808  18 0.004704652
1  112 252 0.307692308
```

3.1.3 Best Prediction among LDA, QDA, Logistic, RF

We next examined the performance of different models. We can see the ROC curves below. ROC curve shows how sensitivity and specificity vary for all possible thresholds and the overall performance of the classifier is summarized by the area under the curve(AUC). The larger the AUC, the better the model performance. In our analysis, the LDA test has outperformed the QDA test. The AUC of the LDA is 94%, the AUC of the QDA is 90%, the AUC of the logistic is 95% and the AUC of the Random Forest is 94%. We found out that Logistic has more area cover than others, hence we preferred the Logistic model.

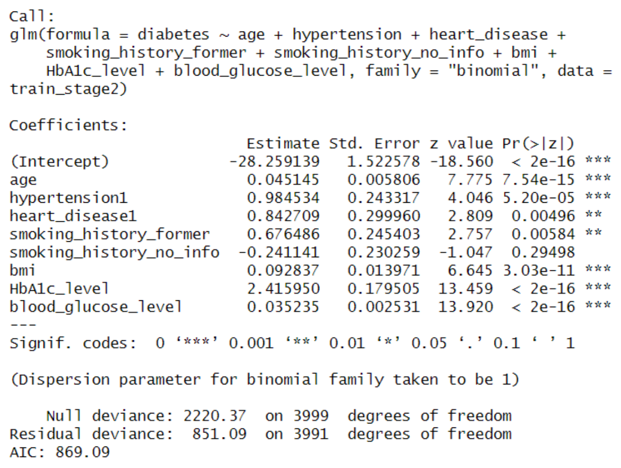


3.2 Stage Two

Instead of taking the means for the missing value for the continuous variable and excluding the categorical variables observations with missing values, we opted for a different strategy in the later part of our analysis, we chose to use iterative regression imputation to handle the missing values and that keeps every observation we have. The notable impact of this change was on the total number of observations, maintaining a more comprehensive dataset could potentially lead to more robust and accurate modeling results. Applying variable selection to this dataset, we observed that the importance variables remained the same as in stage one.

3.2.1 Logistic Regression Prediction

The result from the logistic regression model, as illustrated in the image on the left. It was noticeable that heart disease and smoking history former are less statistically significant to the response variable compared with stage one. The image on the right shows there are no influential observations in this dataset, although there are some high-leverage observations, which is the same as stage one. We test the Deviance and Pearson chi-square test for the model, and both p-values are larger than 0.05, which again shows the model fits the observations well. Lastly, we fit the test set into the model we built for the logistic regression and we get 0.035 for the test error rate which is less than the test error rate in stage one.

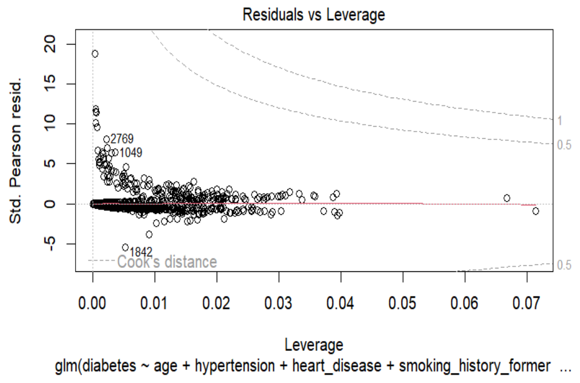


```
Call:
glm(formula = diabetes ~ age + hypertension + heart_disease + 
    smoking_history_former + smoking_history_no_info + bmi + 
    HbA1c_level + blood_glucose_level, family = "binomial", data = 
train_stage2)

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)    
(Intercept)            -28.259139   1.522578 -18.560  < 2e-16 ***
age                      0.045145   0.005806   7.775 7.54e-15 ***
hypertension1            0.984534   0.243317   4.046 5.20e-05 ***
heart_disease1           0.842709   0.299960   2.809  0.00496 ** 
smoking_history_former   0.676486   0.245403   2.757  0.00584 ** 
smoking_history_no_info -0.241141   0.230259  -1.047  0.29498    
bmi                      0.092837   0.013971   6.645 3.03e-11 ***
HbA1c_level              2.415950   0.179505  13.459  < 2e-16 ***
blood_glucose_level      0.035235   0.002531  13.920  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

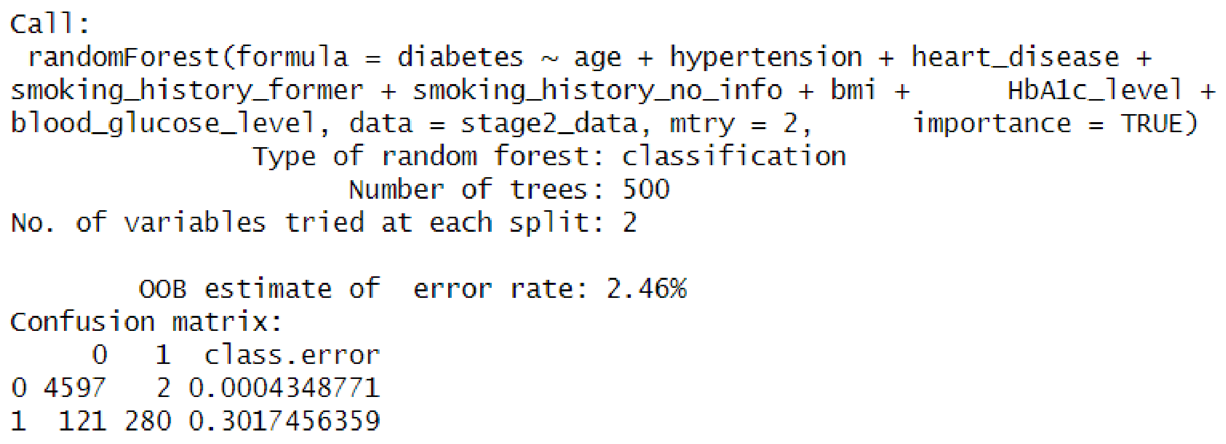
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2220.37  on 3999  degrees of freedom
Residual deviance:  851.09  on 3991  degrees of freedom
AIC: 869.09
```



3.2.2 Random Forest and Bagging

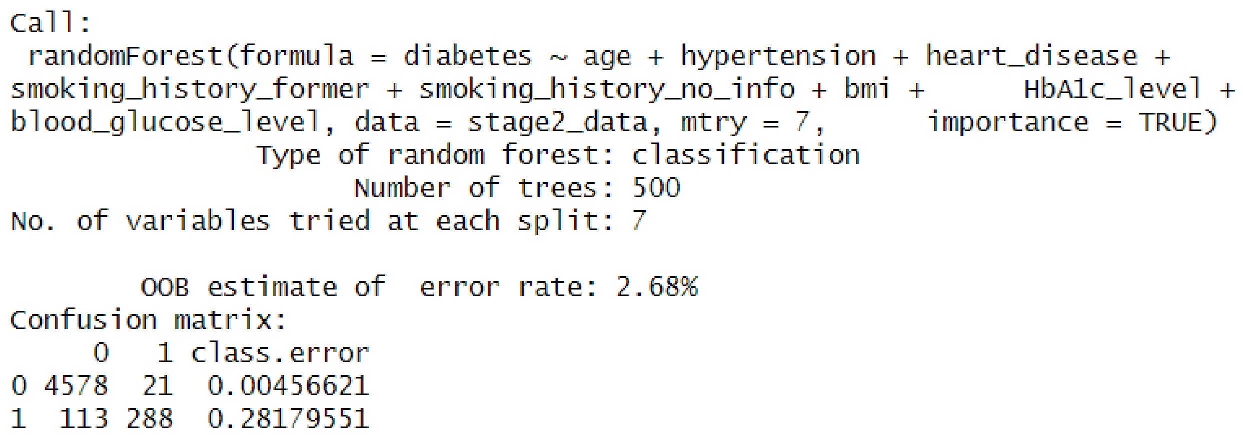
For Random Forest, because there are only 7 predictive variables, we choose the square root of 7 as the number of predictors randomly sampled each time and the number of trees is default setting 500. For Bagging, we consider all 7 predictive variables rather than randomly sampling a few of them.



```
Call:
 randomForest(formula = diabetes ~ age + hypertension + heart_disease + 
smoking_history_former + smoking_history_no_info + bmi +      HbA1c_level + 
blood_glucose_level, data = stage2_data, mtry = 2,      importance = TRUE) 
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 2.46%
Confusion matrix:
     0   1  class.error
0 4597   2 0.0004348771
1  121 280 0.3017456359
```

The table above is the result of Random Forest, the OOB estimate of error rate is 2.46% for Random Forest, and the table below is the result of Bagging and its OOB estimate of error rate is 2.68%. Therefore, the Random Forest is slightly better than Bagging. And the error rate for the testing set is 0.019.

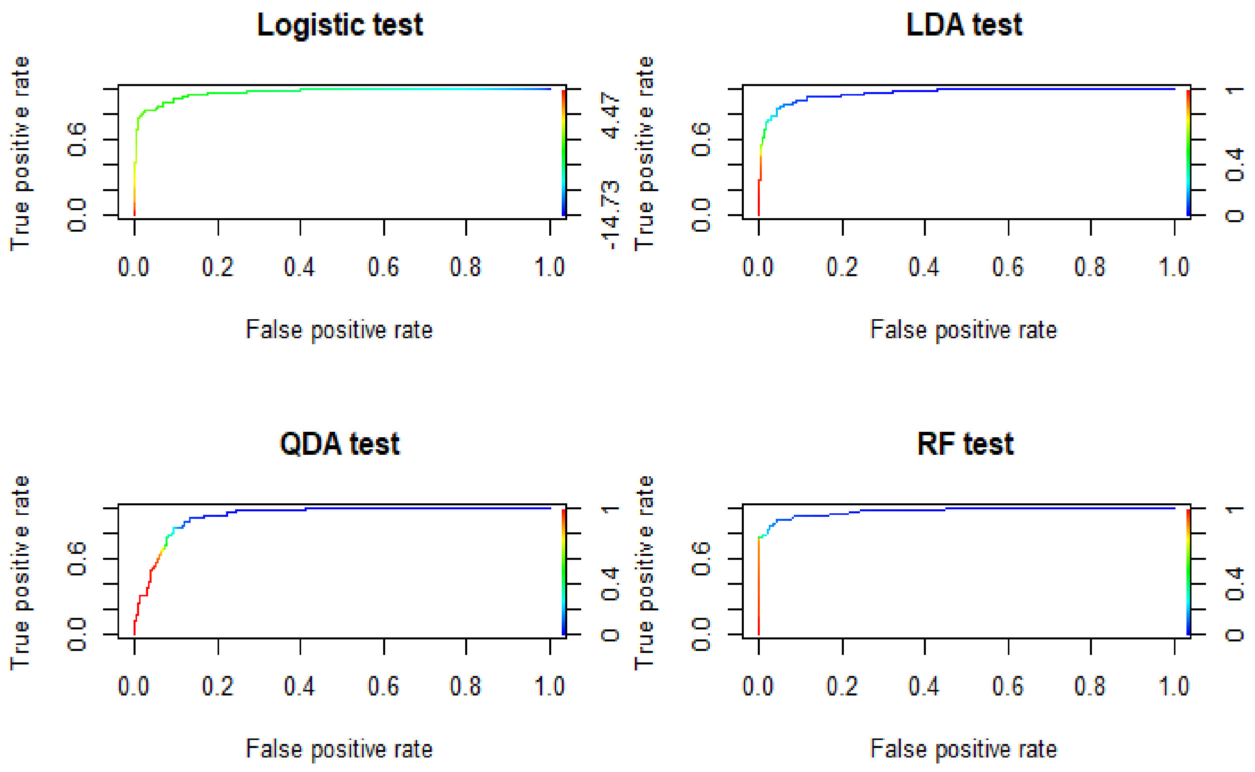


```
Call:
 randomForest(formula = diabetes ~ age + hypertension + heart_disease + 
smoking_history_former + smoking_history_no_info + bmi +      HbA1c_level + 
blood_glucose_level, data = stage2_data, mtry = 7,      importance = TRUE) 
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 7

        OOB estimate of  error rate: 2.68%
Confusion matrix:
     0   1 class.error
0 4578  21  0.00456621
1  113 288  0.28179551
```

3.2.3 Best Prediction among LDA, QDA, Logistic, RF

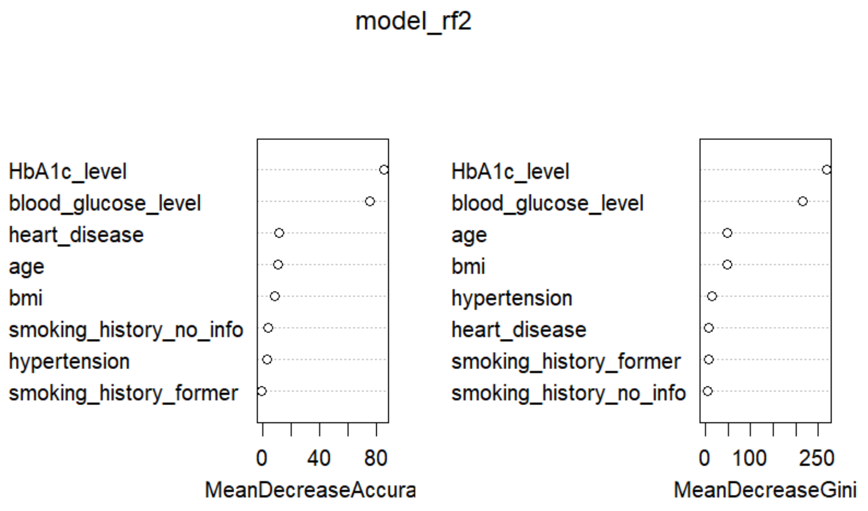
In the following graph, every model performed better because of the different methods of dealing with the missing value. The AUC of the LDA has risen to 97%, the AUC of the QDA has increased to 94%, and both the AUC of the logistic and Random Forest increased to 98%. We found out that the Random Forest model is slightly higher than other models making it the best model of all.



3.3 Test Error Rate Comparison

Names	**Values**
Stage1: Test Error rate for Logistic regression model	0.0501193
Stage1: Test Error rate for LDA	0.0572792
Stage1: Test Error rate for RF	0.0310263
Stage2: Test Error rate for Logistic regression model	0.0350000
Stage2: Test Error rate for LDA	0.0420000
Stage2: Test Error rate for RF	0.0190000

Based on the table above, we can see that Stage 2 has outperformed Stage 1 and the smallest test error rate is Random Forest in Stage 2. I think we can say the more data, the smaller the test error rate will be, and removing observations from the dataset kills the performance of the prediction. If we look at the plot below, which is the result of the Random Forest model in stage two, we can see that the HbA1c level is the most important variable.



4. Discussion and Summary

This study has identified some of the variables as important variables of diabetes, which are age, hypertension, heart disease, smoking history, BMI, HbA1c level, and blood glucose levels, and the HbA1c level is the most important variable. In Logistic regression, the transition from mean replacement for continuous variables and exclusion for categorical variables to iterative regression imputation from stage one to stage two not only keeps the observations but also reduces the test error rate from 0.05 to 0.035. LDA outperformed the QDA and KNN due to its superior ROC curve performance and lower test error rate in both stages and the preferences were consistent even when the dataset size increased in stage 2, highlighting LDA’s robustness. Overall, the best model between stage one and stage two is the Random Forest model in stage two, which has 0.019 test error rate.

When dealing with missing data, different methods can lead to very different results, try multiple ways of replacing missing data might help reduce the test error rate, and perhaps trying different statistical models could help. In this dataset, there are only 8 variables to predict diabetes, in my opinion, there are multiple ways to have diabetes, which might not be included in these 8 variables. Therefore, adding more predictive variables might be able to help improve the decrease the error rate.

Appendix

~~~
# Data import
```{r}
diabetes_data <- read.table("C:\\Users\\54088\\OneDrive\\桌面\\SW\\S5 2023\\STAT 4052\\HW\\Diabetes4.txt")

summary(diabetes_data)
# NA only occurs in hypertension, HbA1c_level, and Diabetes.
# For continuous values impute with mean of that variable.
~~~

~~~~
# For categorical values exclude the observations with missingness from the analysis.
```

# Stage1 - dealing with missing data
```{r}
# stage 1
stage1_diabetes = diabetes_data

# replace the missing value in continuous variables with the mean of that variable
stage1_diabetes$HbA1c_level[is.na(stage1_diabetes$HbA1c_level)] = mean(diabetes_data$HbA1c_level, na.rm =
TRUE)

# remove the observation with missing value in categorical variables
stage1_diabetes <- stage1_diabetes[!is.na(stage1_diabetes$hypertension), ]
stage1_diabetes <- stage1_diabetes[!is.na(stage1_diabetes$diabetes), ]

stage1_diabetes$gender <- as.factor(stage1_diabetes$gender)
stage1_diabetes$smoking_history <- as.factor(stage1_diabetes$smoking_history)

# one-hot code the training and validation set
library(mltools)
library(data.table)
stage1_data <- one_hot(as.data.table(stage1_diabetes), dropUnusedLevels = TRUE)

# factor some variables
stage1_data$hypertension <- as.factor(stage1_data$hypertension)
stage1_data$heart_disease <- as.factor(stage1_data$heart_disease)
stage1_data$diabetes <-as.factor(stage1_data$diabetes)
colnames(stage1_data)[11] <- "smoking_history_no_info"
colnames(stage1_data)[12] <- "smoking_history_not_current"

#since the gender:Other has only one observation left in the dataset after omit the NAs which is in test set in this
case. Therefore, we have to exclude this column.
stage1_data <- subset(stage1_data, select = -c(3))

summary(stage1_data)
```


# Stage1 - logistic regression
```{r}
library(leaps)

set.seed(1234)
train_indices = sample(1:nrow(stage1_data), nrow(stage1_data)*0.8)
train_stage <- stage1_data[train_indices, ]
test_stage <- stage1_data[-train_indices, ]

# Lab5
# Select which model is the best model
# Use stepwise and AIC to determine which model is the optimal model
stepwise_model <- step(glm(diabetes ~ ., data = stage1_data, family = "binomial"), direction = "both", trace = FALSE)
stepwise_model
~~~~

~~~
# Fit the optimal model and predict
model_logistic <- glm(diabetes ~ age + hypertension + heart_disease + smoking_history_former +
smoking_history_no_info + bmi + HbA1c_level + blood_glucose_level, family = "binomial", data = train_stage)
summary(model_logistic)

pre_logistic = predict(model_logistic, newdata = test_stage, type = "response")
pre_logistic_class = ifelse(pre_logistic > 0.5, 1, 0)

table(test_stage$diabetes, pre_logistic_class)
err_logistic_stage1 = mean(test_stage$diabetes!=pre_logistic_class)
err_logistic_stage1

# Diagnostic plot
plot(model_logistic, which = 5)
# There is no influencial points

# Deviance & Pearson chi-square test
pchisq(model_logistic$deviance, 3343, lower.tail = FALSE)

Pearson = sum(residuals(model_logistic, type = "pearson")^2)
pchisq(Pearson, 3343, lower.tail = FALSE)
```

# Stage1 - LDA, QDA, KNN
```{r}
# Sensibly choose either KNN, LDA or QDA to predict diabetes
# Has to explain why LDA is best among these three methods
# Lab9
library(caret)
library(MASS)
# try LDA
model_lda = lda(diabetes ~ age + hypertension + heart_disease + smoking_history_former +
smoking_history_no_info + bmi + HbA1c_level + blood_glucose_level, train_stage)

pre_lda = predict(model_lda, newdata = test_stage)

# Confusion matrix
table(test_stage$diabetes, pre_lda$class)

# Misclassificantion error
err_lda_stage1 <- mean(test_stage$diabetes!=pre_lda$class)
err_lda_stage1

# try QDA
model_qda = qda(diabetes ~ age + hypertension + heart_disease + smoking_history_former +
smoking_history_no_info + bmi + HbA1c_level + blood_glucose_level, train_stage)

pre_qda = predict(model_qda, newdata = test_stage)

# Confusion matrix
table(test_stage$diabetes, pre_qda$class)

# Misclassification error
err_qda_stage1 <- mean(test_stage$diabetes!=pre_qda$class)
~~~

```
err_qda_stage1
# LDA had lower validation error and hence is preferred over QDA

# ROC curve and AUC(LDA and QDA)
library(ROCR)

lda_pre = pre_lda$posterior[,2]
qda_pre = pre_qda$posterior[,2]

pred_lda = prediction(lda_pre, test_stage$diabetes)
perf_lda = performance(pred_lda, "tpr", "fpr")
pred_qda = prediction(qda_pre, test_stage$diabetes)
perf_qda = performance(pred_qda, "tpr", "fpr")
par(mfrow = c(1, 2))
plot(perf_lda, colorize = TRUE, main = "LDA test")
plot(perf_qda, colorize = TRUE, main = "QDA test")

AUC_lda = performance(pred_lda, "auc")@y.values[[1]]
AUC_qda = performance(pred_qda, "auc")@y.values[[1]]
data.frame(model = c("LDA", "QDA"), AUC = c(AUC_lda, AUC_qda))
# In the plot, the LDA is out performed QDA


# KNN unless we find something useful for this one, otherwise we said since LDA has perform better than QDA,
meaning that in this case we can say that it has better fit for linear model like LDA.
library(class)
# try k = 3
pre4 = knn(as.matrix(train_stage$age, train_stage$hypertension, train_stage$heart_disease,
train_stage$smoking_history_former, train_stage$smoking_history_no_info, train_stage$bmi,
train_stage$HbA1c_level, train_stage$blood_glucose_level), as.matrix( test_stage$age, test_stage$hypertension,
test_stage$heart_disease, test_stage$smoking_history_former, test_stage$smoking_history_no_info,
test_stage$bmi, test_stage$HbA1c_level, test_stage$blood_glucose_level), cl = train_stage$diabetes, k = 3)

table(test_stage$diabetes, pre4)
val_err4 = mean(test_stage$diabetes!=pre4)
val_err4

# try k = 5
pre5 = knn(as.matrix(train_stage$age, train_stage$hypertension, train_stage$heart_disease,
train_stage$smoking_history_former, train_stage$smoking_history_no_info, train_stage$bmi,
train_stage$HbA1c_level, train_stage$blood_glucose_level), as.matrix( test_stage$age, test_stage$hypertension,
test_stage$heart_disease, test_stage$smoking_history_former, test_stage$smoking_history_no_info,
test_stage$bmi, test_stage$HbA1c_level, test_stage$blood_glucose_level), cl = train_stage$diabetes, cl =
train_stage$diabetes, k = 5)

table(test_stage$diabetes, pre5)
val_err5 = mean(test_stage$diabetes!=pre5)
val_err5

# try 11
pre6 = knn(as.matrix(train_stage$age, train_stage$hypertension, train_stage$heart_disease,
train_stage$smoking_history_former, train_stage$smoking_history_no_info, train_stage$bmi,
train_stage$HbA1c_level, train_stage$blood_glucose_level), as.matrix( test_stage$age, test_stage$hypertension,
test_stage$heart_disease, test_stage$smoking_history_former, test_stage$smoking_history_no_info,
```

~~~
test_stage$bmi, test_stage$HbA1c_level, test_stage$blood_glucose_level), cl = train_stage$diabetes, cl =
train_stage$diabetes, k = 10)

table(test_stage$diabetes, pre6)
val_err6 = mean(test_stage$diabetes!=pre6)
err_knn_stage1 <- val_err6

# Do ROC curve for LDA, QDA, and KNN. Compare the error rate. and we can see that lda has the minimum error
rate.
data.frame(val_err4, val_err5, val_err6)

```

# Stage1 - Random Forest and Bagging
```{r}
# Use either random forest, bagging or boosting to predict diabetes
# Lab10
# try bagging - classification
# Random forest -> m = sqrt(p) = 2
set.seed(4052)
library(randomForest)
model_rf = randomForest(diabetes ~ age + hypertension + heart_disease + smoking_history_former +
smoking_history_no_info + bmi + HbA1c_level + blood_glucose_level, data = stage1_data, mtry = 2, importance =
TRUE)
model_rf

# Two importance variables are HbA1c_level and blood_glucose_level
# error rate = 0.0316, which is less than LDA

pre_rf <- predict(model_rf, test_stage)
table(pre_rf, test_stage$diabetes)
mean(pre_rf!=test_stage$diabetes)


model_bagging = randomForest(diabetes ~ age + hypertension + heart_disease + smoking_history_former +
smoking_history_no_info + bmi + HbA1c_level + blood_glucose_level, data = stage1_data, mtry = 7, importance =
TRUE)
model_bagging


# Random forest for only training set
model_rf_train = randomForest(diabetes ~ age + hypertension + heart_disease + smoking_history_former +
smoking_history_no_info + bmi + HbA1c_level + blood_glucose_level, data = train_stage, mtry = 2, importance =
TRUE)
model_rf_train

# Two importance variables are HbA1c_level and blood_glucose_level
# error rate = 0.0316, which is less than LDA

pre_rf_train <- predict(model_rf_train, test_stage)
table(pre_rf_train, test_stage$diabetes)
err_rf_stage1 <- mean(pre_rf_train!=test_stage$diabetes)
err_rf_stage1
~~~

~~~
```


# Stage2 - Iterative Regression
```{r}
# iterative regression
stage2_diabetes = diabetes_data

# HbA1c
stage2_diabetes$HbA1c_level[is.na(stage2_diabetes$HbA1c_level)] = mean(diabetes_data$HbA1c_level, na.rm =
TRUE)

# hypertension
stage2_diabetes$hypertension[is.na(stage2_diabetes$hypertension)] = 0

# diabetes
stage2_diabetes$diabetes[is.na(stage2_diabetes$diabetes)] = 0

# Iterative regression
n_iter = 10
for(i in 1:n_iter){
  # impute HbA1c(cont)
  m_HbA1c = lm(HbA1c_level ~ ., stage2_diabetes, subset=!is.na(diabetes_data$HbA1c_level))
  pre_HbA1c = predict(m_HbA1c, stage2_diabetes[is.na(diabetes_data$HbA1c_level),])
  stage2_diabetes$HbA1c_level[is.na(diabetes_data$HbA1c_level)] = pre_HbA1c

  # impute hypertension
  library(nnet)
# impute hypertension (binary)
  m_hypertension = glm(hypertension ~ ., family = binomial, data = stage2_diabetes, subset =
!is.na(diabetes_data$hypertension))
  pre_hypertension = predict(m_hypertension, stage2_diabetes[is.na(diabetes_data$hypertension), ], type =
"response")
  stage2_diabetes$hypertension[is.na(diabetes_data$hypertension)] = ifelse(pre_hypertension > 0.5, 1, 0)

  # impute diabetes
  m_diabetes = glm(diabetes ~., family = binomial, data = stage2_diabetes, subset =
!is.na(diabetes_data$hypertension))
  pre_diabetes = predict(m_diabetes, stage2_diabetes[is.na(diabetes_data$diabetes),], type = "response")
  stage2_diabetes$diabetes[is.na(diabetes_data$diabetes)] = ifelse(pre_diabetes > 0.5, 1, 0)
}

```


# Stage2 - Logistic Regression
```{r}
# stage2
stage2_diabetes$gender <- as.factor(stage2_diabetes$gender)
stage2_diabetes$smoking_history <- as.factor(stage2_diabetes$smoking_history)

# one-hot-encoding
stage2_data <- one_hot(as.data.table(stage2_diabetes), dropUnusedLevels = TRUE)

# factor some variables
~~~

~~~~
stage2_data$hypertension <- as.factor(stage2_data$hypertension)
stage2_data$heart_disease <- as.factor(stage2_data$heart_disease)
stage2_data$diabetes <-as.factor(stage2_data$diabetes)
colnames(stage2_data)[11] <- "smoking_history_no_info"
colnames(stage2_data)[12] <- "smoking_history_not_current"

#since the gender:Other has only one observation left in the dataset after omit the NAs which is in test set in this
case. Therefore, we have to exclude this column.
stage2_data <- subset(stage2_data, select = -c(3))

set.seed(1234)
train_indices = sample(1:nrow(stage2_data), nrow(stage2_data)*0.8)
train_stage2 <- stage2_data[train_indices, ]
test_stage2 <- stage2_data[-train_indices, ]


# Lab5
# Select which model is the best model
# Use stepwise and AIC to determine which model is the optimal model
stepwise_model2 <- step(glm(diabetes ~ ., data = stage2_data, family = "binomial"), direction = "both", trace =
FALSE)
stepwise_model2

# Fit the optimal model and predict
model_logistic2 <- glm(diabetes ~ age + hypertension + heart_disease + smoking_history_former +
smoking_history_no_info + bmi + HbA1c_level + blood_glucose_level, family = "binomial", data = train_stage2)
summary(model_logistic2)

pre_logistic2 = predict(model_logistic2, newdata = test_stage2)
pre_logistic_class2 <- ifelse(pre_logistic2 > 0.5, 1, 0)
table(test_stage2$diabetes, pre_logistic_class2)
err_logistic_stage2 = mean(test_stage2$diabetes!=pre_logistic_class2)
err_logistic_stage2

# Diagnostic plot
plot(model_logistic2, which = 5)

# Deviance & Pearson chi-square test
pchisq(model_logistic2$deviance, 3991, lower.tail = FALSE)

Pearson2 = sum(residuals(model_logistic2, type = "pearson")^2)
pchisq(Pearson2, 3991, lower.tail = FALSE)
```

# Stage2 - LDA, QDA, KNN
```{r}
# Sensibly choose either KNN, LDA or QDA to predict diabetes
# Has to explain why LDA is best among these three methods
# Lab9
str(train_stage2)
library(caret)
library(MASS)
# try LDA
~~~~

```
model_lda2 = lda(diabetes ~ age + hypertension + heart_disease + smoking_history_former +
smoking_history_no_info + bmi + HbA1c_level + blood_glucose_level, train_stage2)

pre_lda2 = predict(model_lda2, newdata = test_stage2)

# Confusion matrix
table(test_stage2$diabetes, pre_lda2$class)

# Misclassificantion error
err_lda_stage2 <- mean(test_stage2$diabetes!=pre_lda2$class)

# try QDA
model_qda2 = qda(diabetes ~ age + hypertension + heart_disease + smoking_history_former +
smoking_history_no_info + bmi + HbA1c_level + blood_glucose_level, train_stage2)

pre_qda2 = predict(model_qda2, newdata = test_stage2)

# Confusion matrix
table(test_stage2$diabetes, pre_qda2$class)

# Misclassification error
err_qda2 <- mean(test_stage2$diabetes!=pre_qda2$class)
# LDA had lower validation error and hence is preferred over QDA

# ROC curve and AUC(LDA and QDA)
library(ROCR)

lda_pre2 = pre_lda2$posterior[,2]
qda_pre2 = pre_qda2$posterior[,2]

pred_lda2 = prediction(lda_pre2, test_stage2$diabetes)
perf_lda2 = performance(pred_lda2, "tpr", "fpr")
pred_qda2 = prediction(qda_pre2, test_stage2$diabetes)
perf_qda2 = performance(pred_qda2, "tpr", "fpr")
par(mfrow = c(1, 2))
plot(perf_lda2, colorize = TRUE, main = "LDA test")
plot(perf_qda2, colorize = TRUE, main = "QDA test")

# In the plot, the LDA is out performed QDA

AUC_lda2 = performance(pred_lda2, "auc")@y.values[[1]]
AUC_qda2 = performance(pred_qda2, "auc")@y.values[[1]]
data.frame(model = c("LDA", "QDA"), AUC = c(AUC_lda2, AUC_qda2))
# In the plot, the LDA is out performed QDA

# KNN unless we find something useful for this one, otherwise we said since LDA has perform better than QDA,
meaning that in this case we can say that it has better fit for linear model like LDA.
library(class)
# try k = 3
pre9 = knn(as.matrix(train_stage2$age, train_stage2$hypertension, train_stage2$heart_disease,
train_stage2$smoking_history_former, train_stage2$bmi, train_stage2$HbA1c_level,
train_stage2$blood_glucose_level), as.matrix(test_stage2$age, test_stage2$hypertension,
test_stage2$heart_disease, test_stage2$smoking_history_former, test_stage2$bmi, test_stage2$HbA1c_level,
test_stage2$blood_glucose_level), cl = train_stage2$diabetes, k = 3)
```

~~~
table(test_stage2$diabetes, pre9)
val_err7 = mean(test_stage2$diabetes!=pre9)
val_err7

# try k = 5
pre10 = knn(as.matrix(train_stage2$age, train_stage2$hypertension, train_stage2$heart_disease,
train_stage2$smoking_history_former, train_stage2$bmi, train_stage2$HbA1c_level,
train_stage2$blood_glucose_level), as.matrix(test_stage2$age, test_stage2$hypertension,
test_stage2$heart_disease, test_stage2$smoking_history_former, test_stage2$bmi, test_stage2$HbA1c_level,
test_stage2$blood_glucose_level), cl = train_stage2$diabetes, k = 5)

table(test_stage2$diabetes, pre10)
val_err8 = mean(test_stage2$diabetes!=pre10)
val_err8

# try 11
pre11 = knn(as.matrix(train_stage2$age, train_stage2$hypertension, train_stage2$heart_disease,
train_stage2$smoking_history_former, train_stage2$bmi, train_stage2$HbA1c_level,
train_stage2$blood_glucose_level), as.matrix(test_stage2$age, test_stage2$hypertension,
test_stage2$heart_disease, test_stage2$smoking_history_former, test_stage2$bmi, test_stage2$HbA1c_level,
test_stage2$blood_glucose_level), cl = train_stage2$diabetes, k = 10)

table(test_stage2$diabetes, pre11)
err_knn_stage2 = mean(test_stage2$diabetes!=pre11)
err_knn_stage2


# Do ROC curve for LDA, QDA, and KNN. Compare the error rate. and we can see that lda has the minimum error
rate.
```



# Stage2 - RF and Bagging
```{r}
# Use either random forest, bagging or boosting to predict diabetes
# Lab10
# try bagging - classification
set.seed(4052)
library(randomForest)
model_rf2 = randomForest(diabetes ~ age + hypertension + heart_disease + smoking_history_former +
smoking_history_no_info + bmi + HbA1c_level + blood_glucose_level, data = stage2_data, mtry = 2, importance =
TRUE)
model_rf2

pre_rf2 <- predict(model_rf2, test_stage2)
table(pre_rf2, test_stage2$diabetes)
mean(pre_rf2!=test_stage2$diabetes)


# Two importance variables are HbA1c_level and blood_glucose_level
# error rate = 0.0316, which is less than LDA
~~~

~~~
model_bagging2 = randomForest(diabetes ~ age + hypertension + heart_disease + smoking_history_former +
smoking_history_no_info + bmi + HbA1c_level + blood_glucose_level, data = stage2_data, mtry = 7, importance =
TRUE)
model_bagging2

pre_bagging2 <- predict(model_bagging2, test_stage2)
table(pre_bagging2, test_stage2$diabetes)
mean(pre_bagging2!=test_stage2$diabetes)


# Random forest for only training set
model_rf_train2 = randomForest(diabetes ~ age + hypertension + heart_disease + smoking_history_former +
smoking_history_no_info + bmi + HbA1c_level + blood_glucose_level, data = train_stage2, mtry = 2, importance =
TRUE)
model_rf_train2

# Two importance variables are HbA1c_level and blood_glucose_level
# error rate = 0.0316, which is less than LDA

pre_rf_train2 <- predict(model_rf_train2, test_stage2)
table(pre_rf_train2, test_stage2$diabetes)
err_rf_stage2 <- mean(pre_rf_train2!=test_stage2$diabetes)

varImpPlot(model_rf2)
```

```{r}
# Comparison between stage1 and stage2
data.frame(
  Names = c("Stage1: Test Error rate for logistic regression model", "Stage1: Test Error rate for LDA", "Stage1: Test
Error rate for RF", "Stage2: Test Error rate for logistic regression model", "Stage2: Test Error rate for LDA", "Stage2:
Test Error rate for RF"),
  Values = c(err_logistic_stage1, err_lda_stage1, err_rf_stage1, err_logistic_stage2, err_lda_stage2, err_rf_stage2))


summary(stage1_data)
summary(stage2_data)
```
~~~