Predicting diabetes mellitus using multiple correspondence analysis and machine learning

Collin Hoskins

Abstract

Machine learning provides many benefits to analyses of health conditions. Machine learning models have been rapidly improving and will continue to improve as more, unbiased data are available to build the improved models. In this brief analysis and study, I measured the performance of three machine learning algorithms in predicting diabetes mellitus in individuals based on multiple predictor variables. The data were split into a training and testing set in a 70/30 split (70% of observations fell in the training set, and 30% of observations fell in the testing set), which is typically the standard for machine learning. I obtained the dataset from the University of California-Irvine Machine Learning Repository (UCIMLR). The data are from direct interviews with patients at Sylhet Diabetes Hospital in Sylhet, Bangladesh. The dataset consists of 520 observations with 17 features (1 numerical, 16 categorical). The numerical feature (age), was converted into an age group feature. Eight features were chosen to train the machine learning models. These features were chosen based on results from a Multiple Correspondence Analysis (MCA) on the data. The selected features were used to train the following models: Logistic Regression, Support Vector Machine Classification, Random Forest Classification, and Naive Bayes Classification. The Random Forest Classification provided the lowest prediction error (5.13%). This model was also cross-validated using the non-exhaustive K-fold Cross-Validation method which provided an accuracy of 95%. All analyses and data preprocessing were completed using The R Project for Statistical Computing (R).

Introduction

Diabetes continues to become increasingly prevalent across different ages and different demographics. In 2018, approximately 10.5% of the adult population of the United States had diabetes[1]. As the prevalance continues to increase, the rate of undetection is also increasing. Of the estimated 34.2 million adults with diabetes in 2018, 7.3 million were undiagonised [1]. The need for efficient and accurate predictive models will continue to grow as artificial intelligence (AI) in general, can be helpful in healthcare decision-making[2]. Aside from the obvious health issues associated with diabetes and the increased risk of death, those diagnosed with diabetes have medical expenditures 2.3 times (2018 data) those without diabetes. The total cost of diabetes in the United States in 2018 was $327 billion[1]. Machine learning can address what symptoms can predict diabetes diagnoses, but it cannot address how to reduce the overall prevalance of diabetes. It is clear that certain lifestyle choices and habits can either decrease or increase the risk of developing diabetes at any age[3].

Objectives

- Find the features that explain the most variance of the response variable
- Use Logistic Regression, Support Vector Machine Classification, and Random Forest Classification as machine learning models
- Find the best machine learning model by comparing test error rates

- Develop a final model that can be used to help doctors predict diabetes without waiting for laboratory test results

Analysis Plan

Exploratory Data Analysis

Exploring this dataset is accomplished mainly through various bar charts with multiple conditions. Since all but one variables are binary, bar charts will provide the most insight because counting the instances of each binary value for each variable provides an oversight of which features have more weight than other features.

Data Analytics Methods

Data preprocessing must occur first in order to complete all proceeding analysis steps. The dataset used in this analysis is already very clean and structured. The only preprocessing that occurred was converting the range of values for the age variable into a categorical feature, in which the categories are "Young", "Mid-Young", "MidAge", "Mid-Old", and "Old".

A **Chi-squared test** is used in this analysis to find variables associated with the observed cases of diabetes that occur due to chance. The resulting p-values from the Chi-squared test indicates the null hypothesis that no relationship exists between each variable and the response variable (diabetes) is rejected, and therefore an association between the variable and the response variable exists[4].

Multiple Correspondence Analysis (MCA) is used to reduce the dimensions of a feature space. More specifically, MCA transforms the categorical data such that many binary columns for each categorical variable in the feature space are created, and only one column is valued at 1. This process increased the dimensions because each categorical variable is matched with multiple of the columns that were created. These values were used to calculate the **Eigenvalue**. For each dimension, the Eigenvalues are corrected using various approaches, but they are then used to calculate the explained interia (variance) in the dataset[5]. Using the calculated interia, a metric known as the **squared cosine** is used to explain the degree of association between certain variables[6].

After utilizing MCA, the best features for the machine learning models are selected, and the dataset is divided into a training set and a testing set. The machine learning models considered are **Support Vector Machine (SVM) Classification**, **Logistic Regression**, **Random Forest (RF) Classification**, and **Naive Bayes (NB) Classification**. The algorithms are trained on the training data, and are tested on the testing data. The testing portion provides predicted values, which can be compared with the actual values. A confusion matrix is used to compare predicted values from both classification models with the actual observed values. The confusion matrix is helpful because it highlights the true positives, false negatives, false positives, and true negatives from the model. Since logistic regression transforms the results into probabilities, the confusion matrix was not used for simplicity in this case.

After each model was formed and used to predict diabetes in individuals, the test error is calculated for each model. The model with the lowest test error was also cross-validated using **K-fold Cross-Validation**. K-fold cross-validation is a non-exhaustive cross-validation method that shuffles and resamples the data, so that each iteration (fold) of the validation process is composed of equal counts of data.

Note: RF classification generally does not require k-fold cross-validation because the data are already reshuffled and resampled during each tree split in the model. However, since a training and testing set is used in this analysis, the k-fold cross-validation further confirms the model chosen was the one with the greatest accuracy.

Data Summary

The dataset used in this analysis has 16 binary variables and 1 numerical variable. The response variable in this dataset is Class, in which the levels are “Positive” and “Negative”. See Figure 1 below for an overview of the dataset[7].

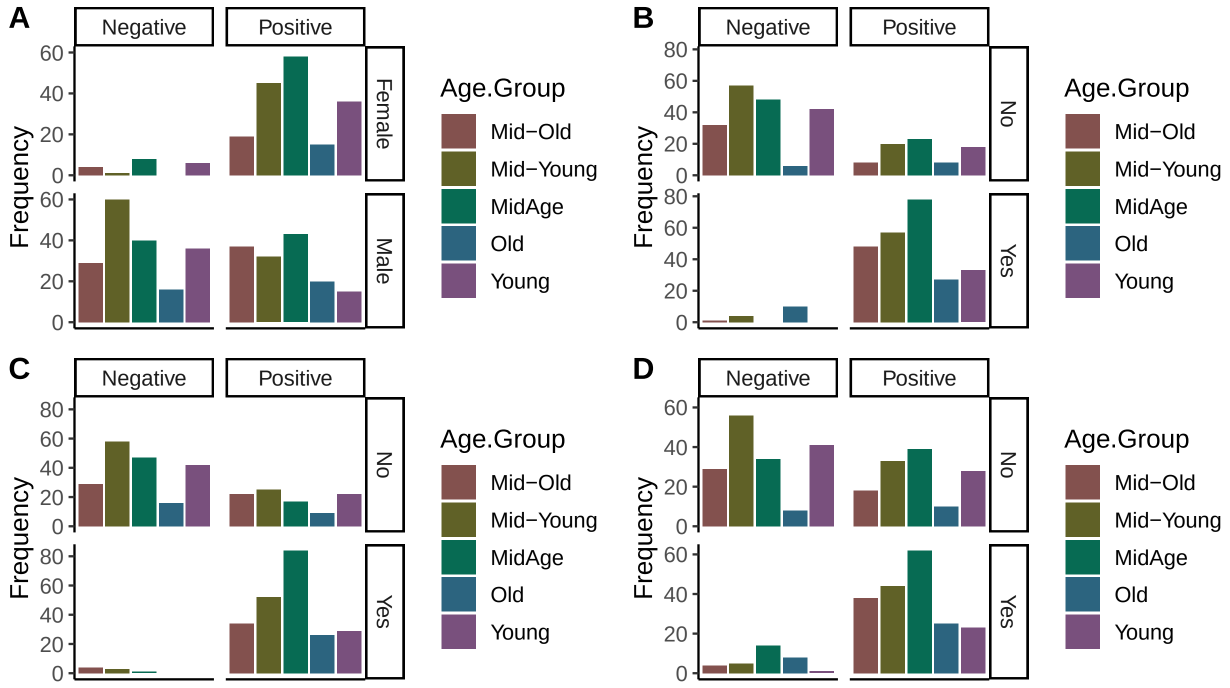

Figure 1: The Number of positive and negative counts for diabetes by age group, as well as sex (A), Polyuria (B), Polydipsia (C), and Partial Paresis (D).

The features that vary across subfigures Figure 1.B, Figure 1.C, and Figure 1.D clearly occur more often in individuals of all ages who are diagnosed positive for diabetes. Another important observation to note is that those who do not have diabetes compose fewer counts than those who tested positive and have the attribute. Also, notice difference between counts of positive cases for those with polyuria (Figure 1.B) and those with polydipsia (Figure 1.C).

Data Analysis and Results

Chi-squared Test

The Chi-squared test revealed that all features except Age group (p.value = 0.12), Itching (p.value = 0.83), and Delayed Healing (p.value = 0.33), do not occur due to chance alone. This gives a general idea of some of the features that have more weight than others in their associatiion with positive or negative diabetes cases. However, because the machine learning models are most accurate when utilizing multiple features, MCA is the best alternative to find the features most closely associated other features.

Multiple Correspondence Analysis (MCA)

Due to the size of the feature selection in this dataset (16), a 2-dimensional feature space is sufficient to find the most closely related features (Figure 2.).

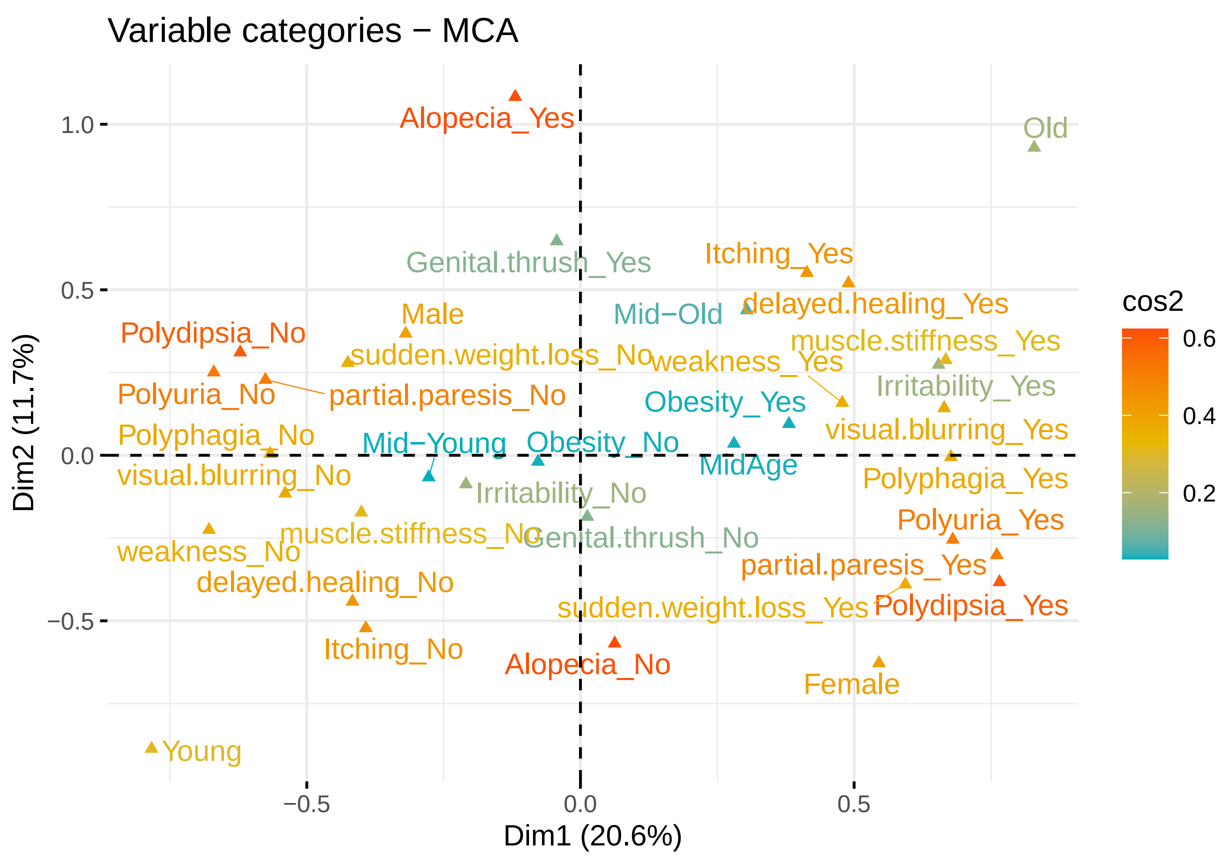

Figure 2: Visual representation of MCA.

The features chosen were selected by observing features with the greatest cos2 value that all occur in dimension 1 (Dim1). The features chosen were: **Polydipsia, Polyphagia, Partial Paresis, Polyuria, Visual Blurring, Delayed Healing, Muscle Stiffness, and Gender.**

Splitting the Dataset

After the features were selected, subsamples of the dataset were selected as the training set and the testing set. I decided to use the standard split ratio of 0.7. The training set consisted of 70% of all observations, and the testing set consisted of 30% of all observations (Table 1.).

Table 1: Length of the training set and testing set.

Training	Testing
364	156

Machine Learning Algorithms

Recall that logistic regression (LogReg), support vector machine (SVM) classification, naive bayes (NB) classification, and random forest (RF) classification were used to find the best predictive model. Logistic regression predicts the probability of each observation in the testing set of having diabetes. The logistic regression revealed many more counts of higher probabilities compared to lower probabilities, but there were many counts near 0% probability, which indicates the model was successful in predicting negative cases of diabetes as well (Figure 3.).

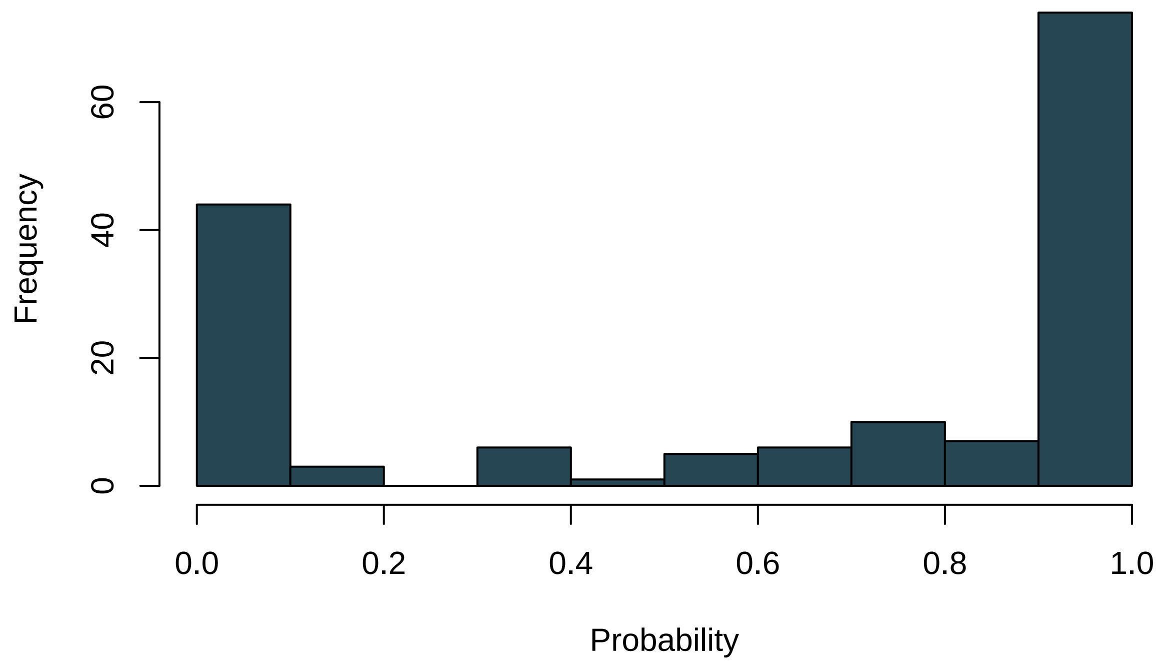

Figure 3: Frequency of probabilities produced from the logistic regression on the testing set.

SVM classification and RF classification produce either "Positive" or "Negative" predictions for diabetes. These results can be compared to the actual values from the testing set by using confusion matrices (Figure 4.).

The bottom right corner of the confusion matrices indicate the true positives. Notice how the SVM classification produced more true positives than the RF classification, but the SVM classification also produced many more false positives than the RF classification produced. This increased the test error for SVM classification (Table 2.). Also, the NB classification model produced false negatives, which increased the test error for this model (Table 2.)

The RF classification algorithm predicted the response variable with the lowest test error of 5.13%, compared to the logistic regression test error of 6.41%, and the SVM classification test error of 10.26% (Table 2.). This indicates the RF classification model is the best model to use to predict diabetes, based on the training dataset on which it was trained. To further depict the accuracy of the model, K-fold cross-validation was used to show the model accuracy across 10 different folds (Table 3.).

Discussion

As mentioned above, the test error for the random forest classification was the lowest out of the three models. The low test error coupled with the 10-fold cross-validation prove this machine learning algorithm was the most effective out of the three chosen at prediciting diabetes (Table 2.). This model could aid doctors in

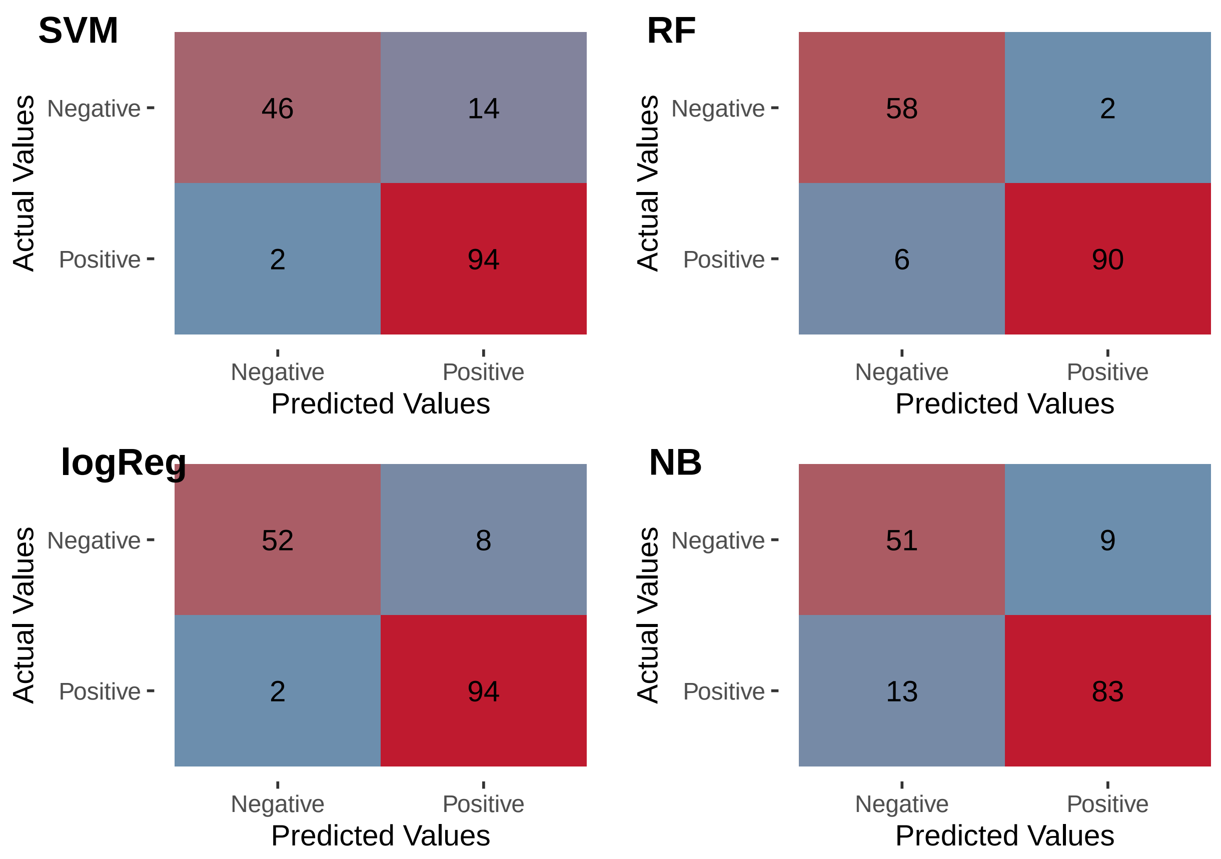

Figure 4: Confusion matrices comparing the actual and predicted values for the support vector machine classification (SVM), random forest classificaiton (RF), logistic regression (LogReg), and Naive Bayes classification (NB).

Table 2: Each Machine learning algorithm and the respective test error for each.

Test	Test Error
LogReg	0.0641026
SVM	0.1025641
RF	0.0512821
NB	0.1410256

Table 3: Accuracy score from the 10-fold cross-validation RF classification.

Accuracy
0.949359

diagnosing diabetes prior to receiving laboratory blood tests, which takes time. In a clinical setting the random forest model could predict diabetes in indvididuals, but the conditions for each feature in the model would have to be met, which is not always the case.

The model could be improved by adding continous variables to the dataset such as blood-glucose levels, etc. These variables would all likely be derived from laboratory tests. In order to achieve this addition, a newly formulated dataset is required. However, a new dataset with continuous variables would likely increase the accuracy of the model, as new feature selection techniques could be implemented, such as Principal

Component Analysis (PCA), which is very common among machine learning practitioners.

For quick assessments, this model is sufficient in diagnosing diabetes. It can, at the least, prompt those who likely have diabetes to undergo blood work to officially receive a diabetes diagnosis. All research objectives were met. However, as mentioned above, a better model can always be constructed given a greater selection of attributes.

Conclusion

This analysis revealed the random forest classification is a sufficient machine learning algorithm to predict diabetes mellitus in individuals. The lack features available in the dataset used for analysis likely reduced the performance of the model. However, this model is sufficient at predicting diabetes mellitus about 95% of the time. The predictive power of this random forest model would increase given a dataset with many more features, especially with the addition of continuous laboratory-produced variables to the dataset.

References

- *[1]:* Statistics About Diabetes | ADA. Diabetes.org. (2018). Retrieved from https://www.diabetes.org/resources/statistics/statistics-about-diabetes.
- *[2]:* Signorini DF, Andrews PJD, Jones PA, et al Predicting survival using simple clinical variables: a case study in traumatic brain injuryJournal of Neurology, Neurosurgery & Psychiatry (1999);66:20-25.
- *[3]:* Pnevmatikakis, Aristodemos, Stathis Kanavos, George Matikas, Konstantina Kostopoulou, Alfredo Cesario, and Sofoklis Kyriazakos. (2021). Risk Assessment for Personalized Health Insurance Based on Real-World Data.Risks 9: 46. https://doi.org/10.3390/risks9030046
- *[4]:* Ling.upenn.edu. (2008). Retrieved from https://www.ling.upenn.edu/~clight/chisquared.htm.
- *[5]:* Abdi, H., & Valentin, D. (2015). Multiple Correspondence Analysis. Researchgate, 1,2,6,8. Retrieved from https://www.researchgate.net/profile/Dominique-Valentin/publication/239542271__Multiple__Correspondence__Analysis/links/54a979900cf256bf8bb95c95/Multiple-Correspondence-Analysis.pdf
- *[6]:* Essentials, M. (2021). MCA - Multiple Correspondence Analysis in R: Essentials - Articles - STHDA. Sthda.com. Retrieved from http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/114-mca-multiple-correspondence-analysis-in-r-essentials/.
- *[7]:* Islam, MM Faniqul, et al. 'Likelihood prediction of diabetes at early stage using data mining techniques.' Computer Vision and Machine Intelligence in Medical Image Analysis. Springer, Singapore, 2020. 113-125

Appendix

R Code Used

```r
# require(dplyr)
# dataset <- read.csv("diabetes_data_upload.csv")
#
# dataset <- mutate(dataset, Age.Group = case_when(Age <= 35 ~ "Young",
#                                                   between(Age, 36, 45) ~ "Mid-Young",
#                                                   between(Age, 46, 55) ~ "MidAge",
#                                                   between(Age, 56, 65) ~ "Mid-Old",
#                                                   Age >= 66 ~ "Old"))
# dataset <- dataset[c("Gender", "Polyuria", "Polydipsia", "sudden.weight.loss",
# "weakness", "Polyphagia", "Genital.thrush",
# "visual.blurring", "Itching", "Irritability",
# "delayed.healing", "partial.paresis", "muscle.stiffness",
# "Alopecia", "Obesity", "Age.Group", "class")]
#
#
# require(FactoMineR)
# require(factoextra)
#
#
#
#
# res.mca <- MCA(dataset[1:16], graph = F)
#
# fviz_mca_var(res.mca, col.var = "cos2",
#              gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
#              repel = TRUE, # Avoid text overlapping
#              ggtheme = theme_minimal())
#
# require(caTools)
#
# set.seed(101)
#
# sample <- sample.split(dataset$class, SplitRatio = 0.7)
#
# train <- subset(dataset, sample == TRUE)
#
# test <- subset(dataset, sample == FALSE)
#
#
# trainSet <-select(train, Polydipsia, Polyphagia, partial.paresis, Polyuria,
# visual.blurring, delayed.healing, muscle.stiffness, Gender, class)
#
# testSet <-  select(test, Polydipsia, Polyphagia, partial.paresis, Polyuria,
# visual.blurring, delayed.healing, muscle.stiffness, Gender, class)
#
# descrData <- data.frame("Training" = length(trainSet[, 1]),
# "Testing" = length(testSet[, 1]))
#
# require(kableExtra)
# kbl(descrData, caption = "Length of the training set and testing set.")
#
# logReg <- glm(factor(class) ~ factor(Polydipsia)  + factor(Polyuria) +
# factor(visual.blurring)
# + factor(Polyphagia) + factor(partial.paresis) +
# factor(delayed.healing) +
#   factor(muscle.stiffness) + factor(Gender) ,
  # family = binomial, data = trainSet)
```

```
#
#
#
# logPredict <- predict(logReg, testSet[-9], type = "response")
#
# hist(logPredict, main = NULL, xlab = "Probability", col = "#264653")
#
# require(e1071)
#
# SVMClass <- svm(factor(class) ~ factor(Polydipsia)  + factor(Polyuria) +
# factor(visual.blurring) + factor(Polyphagia) + factor(partial.paresis) +
# factor(delayed.healing) +    factor(muscle.stiffness) + factor(Gender),
# kernel = "linear", data = trainSet)
#
# SVMPredict <- predict(SVMClass, testSet[-9], type = "response")
#
# require(randomForest)
#
# RFClass <- randomForest(x = trainSet[-9], y = factor(trainSet$class), ntree = 20)
#
# RFPredict <- predict(RFClass, testSet[-9], type = "response")


# require(e1071)
#
#
#
# NBClass <- naiveBayes(x = trainSet[-9], y = factor(trainSet$class))
#
# NBPredict <- predict(NBClass, testSet[-9])

#
# logYPred <- ifelse(logPredict > 0.5, "Positive", "Negative")
#
# logRegError <- 1 - mean(logYPred == testSet$class)
#
# SVMError <- 1 - mean(SVMPredict == testSet$class)
#
# RFError <- 1 - mean(RFPredict == testSet$class)

# NBError <- 1 - mean(NBPredict == testSet$class)

#
# require(ggpubr)
# require(yardstick)
#
#
# Atemp <- conf_mat(table(testSet[, 9], SVMPredict))
#
# Btemp <- conf_mat(table(testSet[, 9], RFPredict))
#
# Ctemp <- conf_mat(table(testSet[, 9], logYPred))
#
```

```
# Dtemp <- conf_mat(table(testSet[, 9], NBPredict))
#
#
# A <- autoplot(Atemp, type = "heatmap") + scale_fill_gradient(low = "#6c8ead",
#                                                               high = "#bf1a2f")+
# xlab("Predicted Values")
# + ylab("Actual Values")
#
# B <- autoplot(Btemp, type = "heatmap") + scale_fill_gradient(low = "#6c8ead",
#                                                               high = "#bf1a2f") +
# xlab("Predicted Values")
# + ylab("Actual Values")
#
# C <- autoplot(Ctemp, type = "heatmap") + scale_fill_gradient(low = "#6c8ead",
#                                                               high = "#bf1a2f") +
# xlab("Predicted Values")
# + ylab("Actual Values")
#
# D <- autoplot(Dtemp, type = "heatmap") + scale_fill_gradient(low = "#6c8ead",
#                                                               high = "#bf1a2f") +
# xlab("Predicted Values")
# + ylab("Actual Values")
#
#
# ggarrange(A, B, C, D, labels = c("SVM", "RF", "logReg", "NB"), nrow = 2, ncol = 2)
#
# ggarrange(A, B, labels = c("SVM", "RF"))
#
# testErrorDf <- data.frame(Test <- c("LogReg", "SVM", "RF"),
# Values <- c(logRegError, SVMError, RFError, NBError))
# kbl(testErrorDf,
# caption = "Each Machine learning algorithm and the respective test error for each.",
#       col.names = c("Test", "Test Error"))
#
# require(caret)
#
# combTestTain <- rbind(trainSet, testSet)
#
#
# train_control <- trainControl(method = "cv", number = 10)
#
# RFcvModel <- train(factor(class) ~ ., data = combTestTain, method = "rf",
# trControl = train_control)
#
# kbl(data.frame(Accuracy = mean(RFcvModel$results[, 2])),
# caption = "Accuracy score from the 10-fold cross-validation RF classification.")
```

Python Code Used to check for Spelling Errors

```
from textblob import TextBlob, Word
import pandas as pd
import numpy as np

import nltk

messages = [line.rstrip() for line in open('spellcheck.txt', encoding = 'utf8')]

words = ''.join(messages)

spellcheck = Word(words)

textblb = TextBlob(words)
textCorrected = textblb.correct()

def compare(text1, text2):
    l1 = text1.split()
    l2 = text2.split()
    good = 0
    bad = 0
    for i in range(0, len(l1)):
        if l1[i] != l2[i]:
            bad += 1
        else:
            good += 1
    return (good, bad)

compare(textblb, textCorrected)
```