Capstone - Choose Your Own Project

Pima Indian Diabetes Database

Fabian Pilz

2023-12-06

Contents

1 Introduction

Diabetes, also known as diabetes mellitus, refers to a group of carbohydrate metabolic disorders that include impaired glucose homeostasis (Bano, 2013). About 1.5 million deaths worldwide are directly attributed to diabetes each year (https://www.who.int/health-topics/diabetes#tab=tab_1). It is also a major cause of blindness, kidney failure heart attacks and strokes. Type 1 diabetes is a chronic autoimmune disease with absolute insulin deficiency. Far more common is type 2 diabetes, usually in adults, which makes up about 90% of the cases worldwide (https://www.diabetesatlas.org/upload/resources/material/20200302_133351_IDFATLAS9e-final-web.pdf). Rates are similar in women and men, with diabetes being the 7th-leading cause of death globally (Murray *et al.*, 2013).

Pima or O'Odham refers to four tribal groups of North American natives in the southwestern United States and northern Mexico, each of whom spoke variants of the "Tepiman/Pima (Pimic) languages". A Pima Indian population near Phoenix, Arizona, has been monitored for diabetes by the National Institute of Diabetes and Digestive and Kidney Disease because of a high incidence rate (Knowler *et al.*, 1981) (Smith *et al.*, 1988). This dataset is a subset of the original larger database and contains only female patients who are at least 21 years old and is available at Kaggle (https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database). The dataset consists of one target variable, 'Outcome', and several predictor variables such as glucose level, age, BMI, etc.

The objective of the project was to predict diabetes diagnoses accurately based on the diagnostic measures using machine learning algorithms. Since we have only two possible outcomes - diabetes or no diabetes - we are dealing with a classification problem.

This report starts with an exploratory data analysis, followed by the an overview of the applied evaluation metrics. Different techniques to deal with classification problems are explained briefly and their application on the training data is presented in the Results section. The relative performance of the differenct models will be discussed afterwards followed by a conclusion which focuses on the limitations of the applied models, the data itself and closes with opportunies for future work.

2 Evaluation metrics

To evaluate the performance of the different machine learning algorithms we first have to define evaluation metrics. We use:

- the *harmonic* average F_1 score,
- overall accuracy,
- Cohen's kappa,
- sensitivity,
- specificity.

To get all of these parameters for a model at once, we define a suitable function.

```r
# define harmonic average, because we have a
# classification problem; F_meas function of
# caret package get params from confusion
# matrix confusionMatrix(data =
# predicted_outcomes, reference =
# diabetes_validation$outcome)
confusionMatrix_params <- function(data, reference) {
    f_meas <- F_meas(data = data, reference = reference)
    confusion_matrix <- confusionMatrix(data = data,
        reference = reference)
    accuracy <- confusion_matrix$overall["Accuracy"]
    kappa <- confusion_matrix$overall["Kappa"]
    other_params <- confusion_matrix$byClass[c("Sensitivity",
        "Specificity", "Pos Pred Value")]
    params <- tibble(f_value = f_meas, accuracy = accuracy,
        kappa = kappa, sensitivity = other_params[1],
        specificity = other_params[2], precision = other_params[3])
    return(params)
}
```

In the case of two-class classification problems, there are four possible prediction outcomes.

	Actual Positive	Actual Negative
Predicted Positive	True Positive	False Positive
Predicted Negative	False Negative	True Negative

Specificity asks: "Out of all subjects that do not have the disease, how many got negative results?"

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}$$

Precision represents the true positive fraction of all positive predictions.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

On the other hand, recall is the fraction of positives that were retrieved. Recall can also be called sensitivity or true positive rate.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

The F_1 score is a way to combine precision and recall the following way:

$$F_1 = 2 * \frac{\text{Precision} \;\times\; \text{Recall}}{\text{Precision} + \text{Recall}}$$

A classifier can only have a high F_1 score, also called *harmonic* average, if it has both high precision and high recall. Therefore, we ultimately use the F_1 score to select the final prediction model. Additionally, we report Cohen's kappa (κ) coefficient which is a statistic that is used to measure inter-rater reliability for categorical items. In the case of binary classifications the formula is the following:

$$\kappa = \frac{2 \times (\text{TP} \times \text{TN} - \text{FN} \times \text{FP})}{(\text{TP} + \text{FP}) \times (\text{FP} + \text{TN}) + (\text{TP} + \text{FN}) \times (\text{FN} + \text{TN})}$$

where TP are the true positives, FP are the false positives, TN are the true negatives, and FN are the false negatives.

3 Data Import

The data can be downloaded directly form an open source (https://datahub.io/machine-learning/diabetes/datapackage.json). For easier access, the data has been downloaded as a .csv file from Kaggle and saved in the project folder. It is imported using the following code:

```r
# import csv file
diabetes_df_raw <- rio::import(here("diabetes.csv"))
```

4 Exploratory Data Analysis and Wrangling

4.1 Overview

To get a better understanding of the dataset, we take a look at it and print the summary statistics.

```
# structure
glimpse(diabetes_df_raw)
```

```
## Rows: 768
## Columns: 9
## $ Pregnancies              <int> 6, 1, 8, 1, 0, 5, 3, 10, 2, 8, 4, 10, 10, 1, ~
## $ Glucose                  <int> 148, 85, 183, 89, 137, 116, 78, 115, 197, 125~
## $ BloodPressure            <int> 72, 66, 64, 66, 40, 74, 50, 0, 70, 96, 92, 74~
## $ SkinThickness            <int> 35, 29, 0, 23, 35, 0, 32, 0, 45, 0, 0, 0, 0, ~
## $ Insulin                  <int> 0, 0, 0, 94, 168, 0, 88, 0, 543, 0, 0, 0, 0, ~
## $ BMI                      <dbl> 33.6, 26.6, 23.3, 28.1, 43.1, 25.6, 31.0, 35.~
## $ DiabetesPedigreeFunction <dbl> 0.627, 0.351, 0.672, 0.167, 2.288, 0.201, 0.2~
## $ Age                      <int> 50, 31, 32, 21, 33, 30, 26, 29, 53, 54, 30, 3~
## $ Outcome                  <int> 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, ~
```

```
# summary
summary(diabetes_df_raw)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin          BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##     Outcome
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.349
##  3rd Qu.:1.000
##  Max.   :1.000
```

The dataset consists of 768 observations of 9 variables. These are:

- pregnancies,
- glucose,
- blood pressure,
- skin thickness,
- insulin,
- BMI,
- diabetes pedigree function,
- age, and
- outcome.

Intrestingly, the summary statistics show many zero values for some of the variables. In some cases we have so many zero values that even the first quantile is affected, e.g. in the case of insulin. After some changes in the data frame, we check how many zeros we have for each potential predictor.

```r
# tidy column names, outcome not as factor of numeric data (problems in model_building
diabetes_df_clean <- diabetes_df_raw %>%
  janitor::clean_names() %>%
  mutate(# id = 1:nrow(diabetes_df),
         outcome = if_else(outcome == '0', 'no', 'yes')) %>%
  mutate(outcome = factor(outcome)) %>%
  rename(diabetes_p_fct = diabetes_pedigree_function) %>%
  mutate(across(where(is.integer), as.double))
```

```r
# check how many zero we have in each column
diabetes_df_clean %>%
    mutate_if(is.numeric, ~(. == 0)) %>%
    dplyr::select(-outcome) %>%
    colSums() %>%
    print()
```

```
##    pregnancies        glucose blood_pressure skin_thickness        insulin
##            111              5             35            227            374
##            bmi diabetes_p_fct            age
##             11              0              0
```

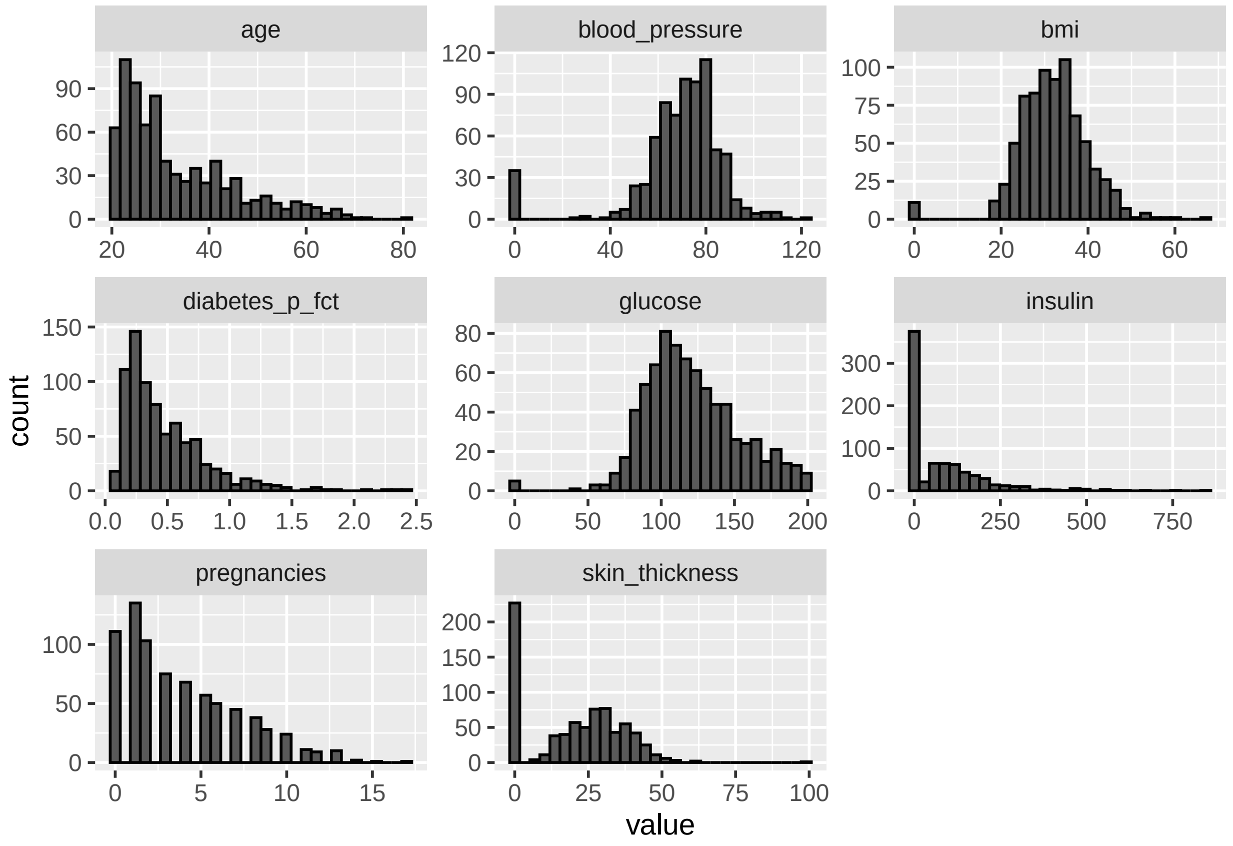
We are missing data for some variables. In the case of insulin, there is no data available for 374 patients.

```r
# percentage of NAs in insulin variable
na_percentage <- 374/nrow(diabetes_df_clean) *
    100
print(na_percentage)
```

```
## [1] 48.69792
```

This affects 48.7 % of all patients and will influence our machine learning models markedly.

```r
# plot all variables for diabetes_data_clean
diabetes_df_clean %>%
    pivot_longer(cols = -outcome, names_to = "param") %>%
    ggplot(aes(x = value)) + geom_histogram(color = "black") +
    facet_wrap(~param, scales = "free")
```



The boxplots visualize the impact these missing values have on the normal distribution of the variables blood pressure, body mass index, glucose, skin thickness, and especially insulin.

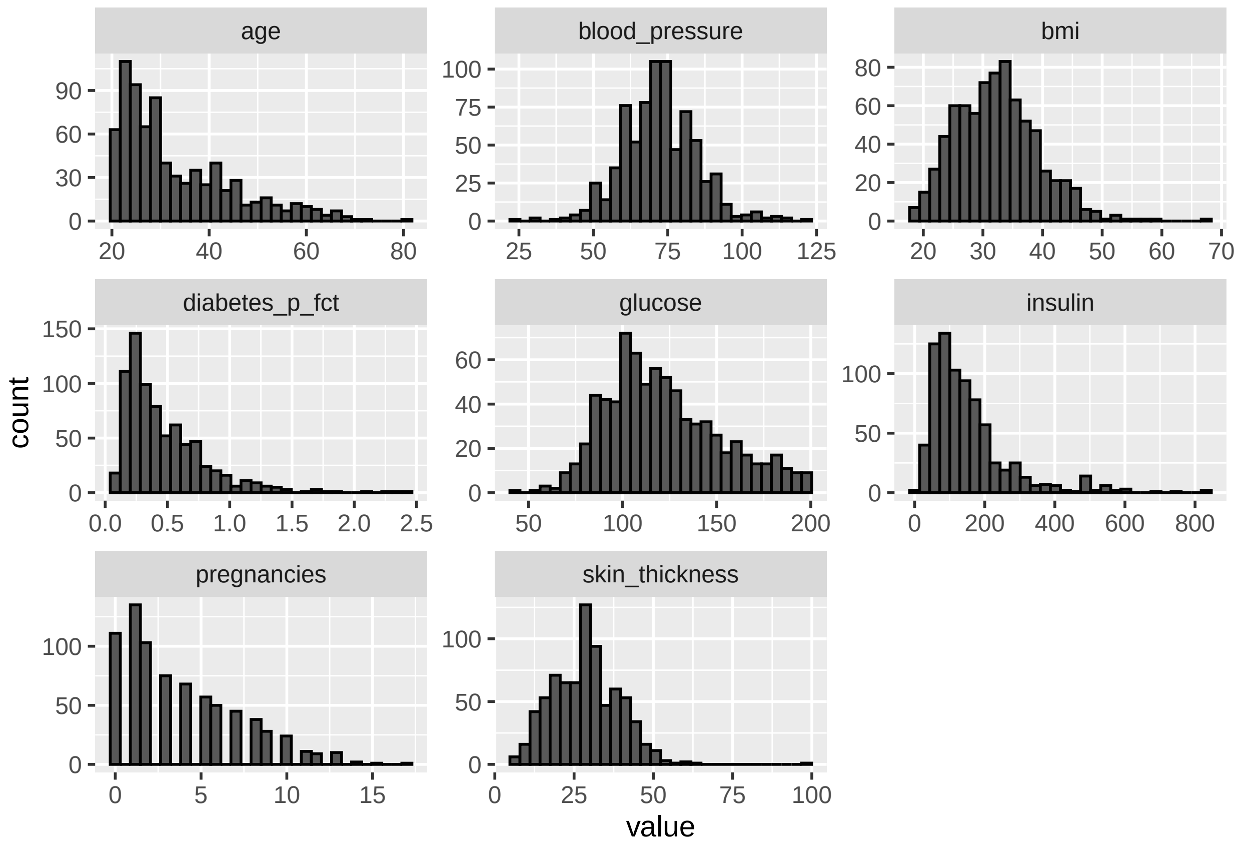
4.2 Data Pre-processing

Therefore, we are going to impute the missing data with values generated by the `mice` package using Fully Conditional Specification implemented by the MICE algorithm as described by Van Buuren and Groothuis-Oudshoorn in 2011 (Van Buuren & Groothuis-Oudshoorn, 2011).

```r
# which columns to modify? ->
# blood_pressure, bmi, glucose, insulin,
# skin_thickness
variables_to_adjust <- c("blood_pressure", "bmi",
    "glucose", "insulin", "skin_thickness")
# replace zero values in relevant variables
# with NA
diabetes_df <- diabetes_df_clean
for (i in seq_along(variables_to_adjust)) {
    diabetes_df[, variables_to_adjust[i]][diabetes_df[,
        variables_to_adjust[i]] == 0] <- NA
}
# replace NA with model data generated by
# mice package
mice_mod <- mice(diabetes_df[, variables_to_adjust],
    method = "rf", seed = 1234, printFlag = FALSE)
diabetes_df[, variables_to_adjust] <- complete(mice_mod)
```

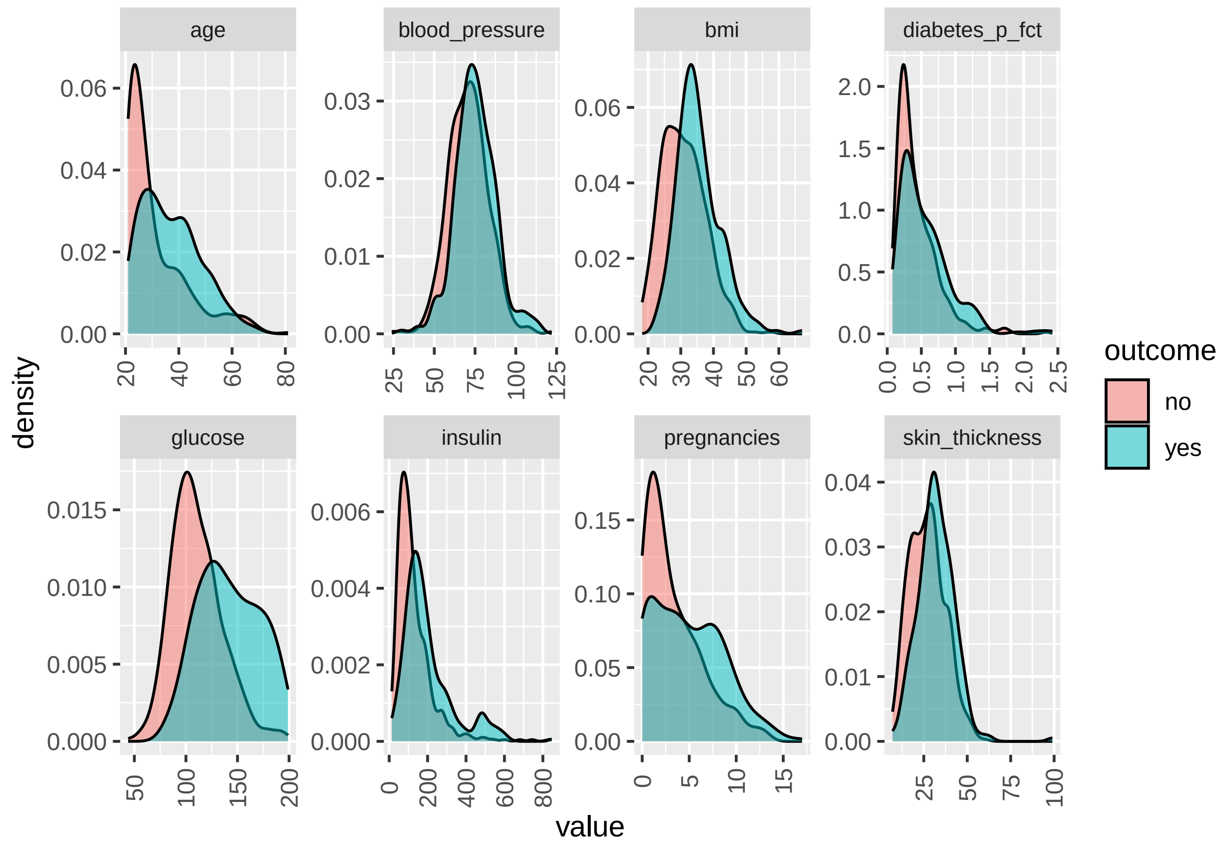
The affected variables look now normally distributed.

```r
# plot all variables for diabetes_data with
# modified dataset
diabetes_df %>%
    pivot_longer(cols = -outcome, names_to = "param") %>%
    ggplot(aes(x = value)) + geom_histogram(color = "black") +
    facet_wrap(~param, scales = "free")
```



Now we can get a first idea of which variables have a higher impact on the outcome than others.

```
# density plots for each param depending on
# the outcome
diabetes_df %>%
    pivot_longer(cols = -outcome, names_to = "variable",
        values_to = "value") %>%
    ggplot(aes(value)) + geom_density(aes(fill = outcome),
    alpha = 0.5) + theme(axis.text.x = element_text(angle = 90,
    vjust = 0.5, hjust = 0.5), strip.text = element_text(size = 8)) +
    facet_wrap(~variable, scales = "free", nrow = 2)
```



The density plot reveals that patients with diabetes tend to have higher levels of glucose and insulin while others, e.g. like blood pressure, have seemingly no impact.

4.3 Split the data set into training, validation and test set

Now we split the dataset into three different subsets in the ratio 80/10/10. The largest subset will be used for training the model and one of the smaller ones to validate the model performance on new data. Because the ultimate goal of a machine learning algorithm is to perform with completely new datasets the third subset will be used to test the performance of the final model.

```
# split data into train, validation and test
# sets Final hold-out test set will be 10%
# of MovieLens data
set.seed(1, sample.kind = "Rounding")  # if using R 3.6 or later
test_index <- createDataPartition(y = diabetes_df$outcome,
    times = 1, p = 0.2, list = FALSE)
diabetes_train <- diabetes_df[-test_index, ]
temp <- diabetes_df[test_index, ]

# split temp into validation and test sets
```

```r
test_index <- createDataPartition(y = temp$outcome,
    times = 1, p = 0.5, list = FALSE)
diabetes_validation <- temp[-test_index, ]
diabetes_test <- temp[test_index, ]

# rm unnecessary data
rm(test_index, temp)

# inspect train set
table(diabetes_train$outcome)
```

The training set is unbalanced and contains predominantly healthy subjects.

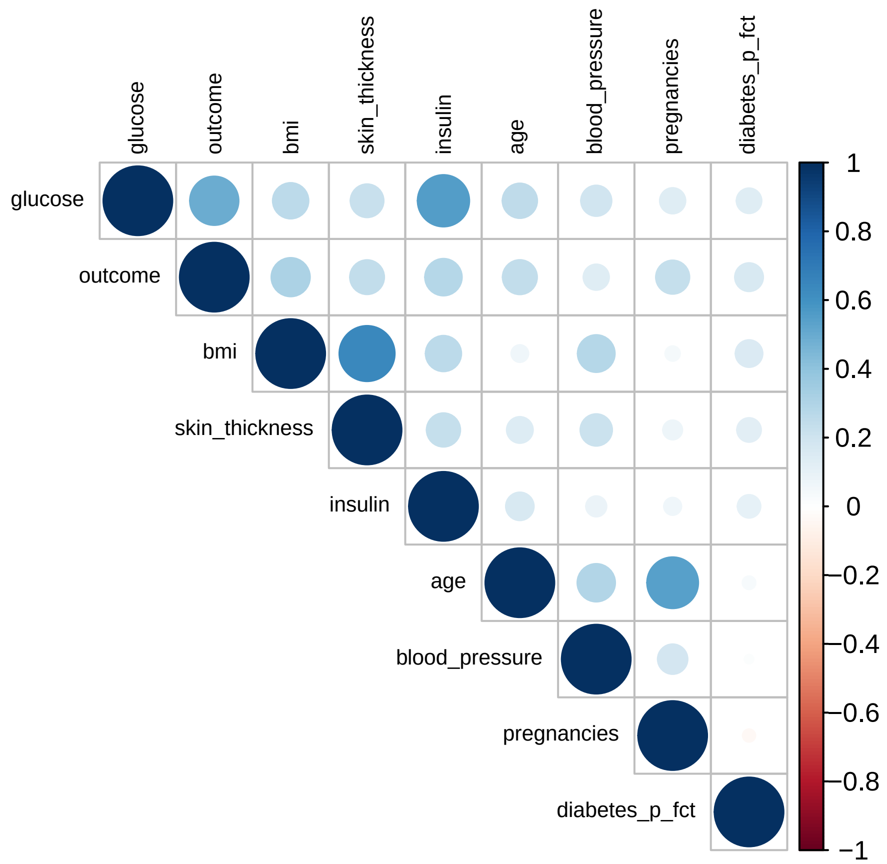
4.4 Correlation between variables

We can now inspect the relationship between variables in the training set.

```r
# create matrix of diabetes_train
corr_df <- diabetes_train %>%
    mutate(outcome = as.factor(ifelse(outcome ==
        "no", "0", "1")))

# analyze the correlation (Pearson's),
# include p-values; use Hmisc package
res <- rcorr(as.matrix(corr_df))

# plot the correlation of variables
corrplot(res$r, type = "upper", tl.cex = 0.7,
    order = "FPC", tl.col = "black")
```



As already shown in the density plots, glucose and insulin levels are highly positively correlated with a diabetes diagnosis. Apart from blood pressure, all other variables also appear to have a positive influence. However, none of the correlations is greater than 0.75. With the exception of blood pressure, we cannot exclude any of the variables.

5 Method development

Apart from one explanatory model, classification trees, we are going to use predictive models and train them with the entire `diabetes_train` set. We define the control parameters for the `train()` function the following way:

```r
# define fit for control function
fit_control <- trainControl(method = "cv", number = 5,
    classProbs = TRUE, summaryFunction = twoClassSummary)
```

5.1 Model 1: Guessing

We choose weighted guessing as our baseline classifier. In this case we just ignore the medical predictors and guess at the weighted percentages of each class.

```r
# percentage of positive diagnoses
ratio <- diabetes_train$outcome %>%
    as.character() %>%
    str_replace_all(., c(no = "0", yes = "1")) %>%
    as.numeric() %>%
    mean()

# build a model
set.seed(1, sample.kind = "Rounding")
predicted_outcomes <- diabetes_validation %>%
    mutate(guess = sample(c(0, 1), size = n(),
        replace = TRUE, prob = c((1 - ratio),
            ratio))) %>%
    mutate(guess = as.factor(str_replace_all(guess,
        c(`0` = "no", `1` = "yes")))) %>%
    pull(guess)

# evaluation metrics
params_model_1 <- confusionMatrix_params(data = predicted_outcomes,
    reference = diabetes_validation$outcome)
```

5.2 Model 2: Logistic Regression

Logistic regression is limited to only two-class classification problems. We predict the categorical dependent variable `outcome` using all independent variables.

```r
# set the seed
set.seed(1, sample.kind = "Rounding")

# do linear model with caret package glm
model_2 <- caret::train(outcome ~ ., method = "glm",
    family = "binomial", metric = "ROC", tuneLength = 10,
    preProcess = c("center", "scale"), trControl = fit_control,
    data = diabetes_train)

# predict outcomes using model 2
predicted_outcomes <- predict(model_2, diabetes_validation)

# evaluation metrics
params_model_2 <- confusionMatrix_params(data = predicted_outcomes,
    reference = diabetes_validation$outcome)
```

5.3 Model 3: Random Forest

A random forest is a classification and regression method that consists of several uncorrelated decision trees. All decision trees are grown under a certain type of randomization during the learning process. The individual trees are then combined to form an ensemble, the Random Forest.

```r
# set the seed
set.seed(1, sample.kind = "Rounding")

# train the model
model_3 <- train(outcome ~ ., diabetes_train,
    method = "rf", tuneLength = 2, trControl = fit_control)

# predict outcomes using model 3
predicted_outcomes <- predict(model_3, diabetes_validation)

# evaluation metrics
params_model_3 <- confusionMatrix_params(data = predicted_outcomes,
    reference = diabetes_validation$outcome)
```

5.4 Model 4: Fitting XGBoost

XGBoost is short for "eXtreme Gradient Boosting" and an open-source software library. This method is based on decision trees and represents an improvement on other methods such as random forest and gradient boosting.

```r
# create tuning grid
xgb_grid = expand.grid(nrounds = 50, eta = c(0.03),
    max_depth = 1, gamma = 0, colsample_bytree = 0.6,
    min_child_weight = 1, subsample = 0.5)

# create model
set.seed(1, sample.kind = "Rounding")
model_4 <- train(outcome ~ ., diabetes_train,
    method = "xgbTree", metric = "ROC", tuneGrid = xgb_grid,
    trControl = fit_control)

# predict outcomes using model 4
predicted_outcomes <- predict(model_4, diabetes_validation)

# evaluation metrics
params_model_4 <- confusionMatrix_params(data = predicted_outcomes,
    reference = diabetes_validation$outcome)
```

5.5 Model 5: K-nearest neighbors

The K-nearest neighbor (kNN) algorithm, is a nonparametric supervised learning classifier that uses the concept of proximity to make classifications or predictions about the grouping of a single data point. It is based on the assumption that similar points can be found in proximity to each other.

```r
# tune k, perform cross-validation
set.seed(3, sample.kind = "Rounding")
model_5 <- train(outcome ~ ., diabetes_train,
    method = "knn", metric = "ROC", trControl = fit_control,
    tuneGrid = expand.grid(k = seq(1, 101, 2)))

# predict outcomes using model 5
predicted_outcomes <- predict(model_5, diabetes_validation)

# evaluation metrics
params_model_5 <- confusionMatrix_params(data = predicted_outcomes,
    reference = diabetes_validation$outcome)
```

5.6 Model 6: Naive Bayes

The Naive Bayes classifier is a supervised machine learning model based on Bayes' Theorem with the "naive" assumption of conditional independence between every pair of features. The approach is mathematically similar to the logistic regression prediction.

```r
# set up tuning grid
search_grid <- expand.grid(usekernel = c(TRUE,
    FALSE), fL = 0:5, adjust = seq(0, 5, by = 1))

# train model
set.seed(3, sample.kind = "Rounding")
model_6 <- train(outcome ~ ., diabetes_train,
    method = "nb", trControl = fit_control, tuneGrid = search_grid,
    preProc = c("BoxCox", "center", "scale", "pca"))

# predict outcomes using model 6
predicted_outcomes <- predict(model_6, diabetes_validation)

# evaluation metrics
params_model_6 <- confusionMatrix_params(data = predicted_outcomes,
    reference = diabetes_validation$outcome)
```

5.7 Model 7: Decision Tree

Classification trees, or decision trees, are another approach to predict the outcome in classification and regression problems. Predictions are formed by calculating which class is the most common among the training set observations. In the flow-chart like structure, each node represents a "test" on a variable which are connected by branches.

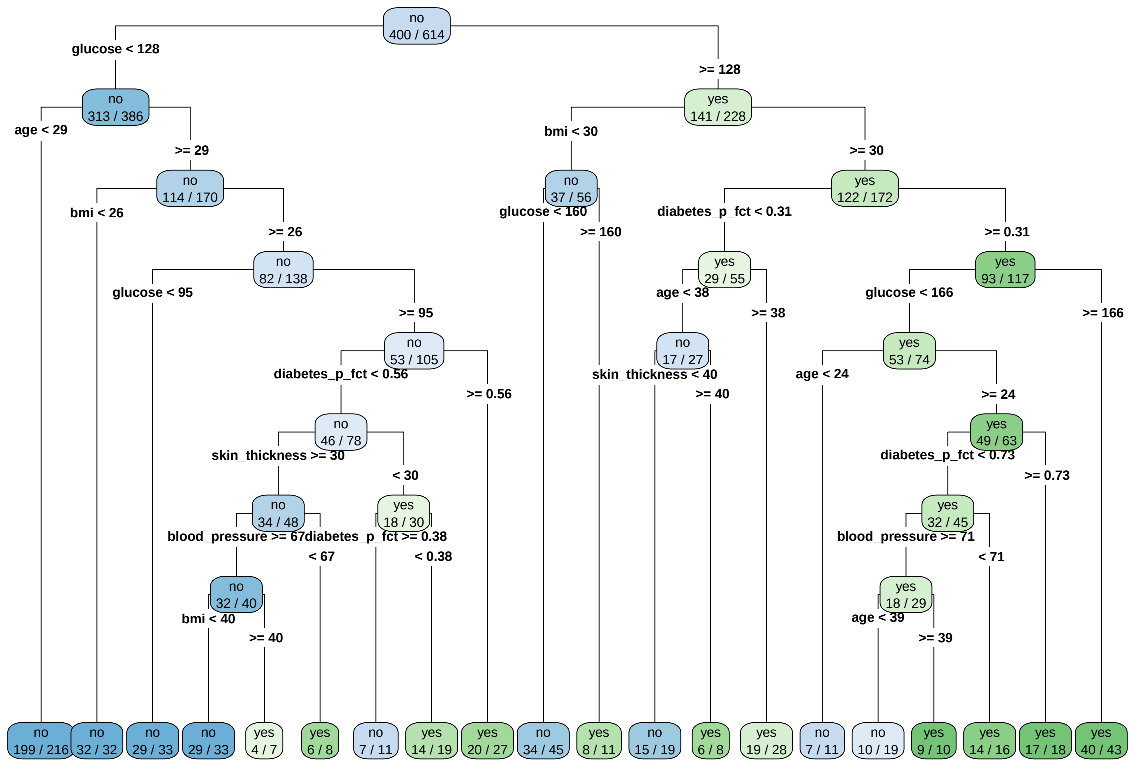
```r
# train model
set.seed(3, sample.kind = "Rounding")
model_7 <- train(outcome ~ ., diabetes_train,
    method = "rpart", metric = "ROC", tuneLength = 20,
    trControl = fit_control)

# predict outcomes using model 7
predicted_outcomes <- predict(model_7, diabetes_validation)

# evaluation metrics
params_model_7 <- confusionMatrix_params(data = predicted_outcomes,
    reference = diabetes_validation$outcome)
```

In the resulting decision tree subjects with high glucose level, high body mass index and of advanced age are more likely to be classfied as diabetic.

```r
# plot
rpart.plot(model_7$finalModel, type = 4, extra = 2,
    cex = 0.45, fallen.leaves = TRUE)
```


no
400 / 614
glucose < 128
>= 128
no
313 / 386
age < 29
>= 29
yes
141 / 228
bmi < 30
>= 30
no
114 / 170
bmi < 26
>= 26
no
37 / 56
glucose < 160
>= 160
yes
122 / 172
diabetes_p_fct < 0.31
>= 0.31
no
82 / 138
glucose < 95
>= 95
yes
29 / 55
age < 38
>= 38
yes
93 / 117
glucose < 166
>= 166
no
53 / 105
diabetes_p_fct < 0.56
>= 0.56
no
17 / 27
skin_thickness < 40
>= 40
yes
53 / 74
age < 24
>= 24
no
46 / 78
skin_thickness >= 30
< 30
yes
49 / 63
diabetes_p_fct < 0.73
>= 0.73
no
34 / 48
blood_pressure >= 67
< 67
yes
18 / 30
diabetes_p_fct >= 0.38
< 0.38
yes
32 / 45
blood_pressure >= 71
< 71
no
32 / 40
bmi < 40
>= 40
yes
18 / 29
age < 39
>= 39
no 199 / 216
no 32 / 32
no 29 / 33
no 29 / 33
yes 4 / 7
yes 6 / 8
no 7 / 11
yes 14 / 19
yes 20 / 27
no 34 / 45
yes 8 / 11
no 15 / 19
yes 6 / 8
yes 19 / 28
no 7 / 11
no 10 / 19
yes 9 / 10
yes 14 / 16
yes 17 / 18
yes 40 / 43

6 Results and Discussion

The following table provides an overview of the key-performance characteristics of all seven models.

```
# bind evaluation params
eval_params <- rbind(params_model_1, params_model_2,
    params_model_3, params_model_4, params_model_5,
    params_model_6, params_model_7) %>%
    mutate(`Model No.` = c(1:7), `Model name` = c("Guess",
        "glm", "RF", "XGBoost", "kNN", "NB", "rpart"),
        .before = f_value)
knitr::kable(eval_params, digits = 5)
```

Model No.	Model name	f_value	accuracy	kappa	sensitivity	specificity	precision
1	Guess	0.55102	0.42857	-0.23379	0.54	0.22222	0.56250
2	glm	0.84615	0.79221	0.52761	0.88	0.62963	0.81481
3	RF	0.83495	0.77922	0.50247	0.86	0.62963	0.81132
4	XGBoost	0.82883	0.75325	0.40213	0.92	0.44444	0.75410
5	kNN	0.85455	0.79221	0.50121	0.94	0.51852	0.78333
6	NB	0.78431	0.71429	0.36172	0.80	0.55556	0.76923
7	rpart	0.81633	0.76623	0.49527	0.80	0.70370	0.83333

As expected, simple guessing delivers by far the worst results. Based on the F_1-score, the kNN approach performs best. It has also the second highest value for κ. The performance of the Random Forests model is comparable. We now merge the `diabetes_train` data set with the `diabetes_validation` set and use the combined data to retrain our final model utilizing the kNN algorithm.

```
### do it for the final test set merge
### diabetes_train and diabetes_validation
diabetes_df_final <- full_join(diabetes_train,
    diabetes_validation)

# train the final model, therefore tune k,
# perform cross-validation
set.seed(2, sample.kind = "Rounding")
model_final <- train(outcome ~ ., diabetes_df_final,
    method = "knn", metric = "ROC", trControl = fit_control,
    tuneGrid = expand.grid(k = seq(1, 101, 2)))

# predict outcomes for diabetes_test using
```

```r
# model_final
predicted_outcomes <- predict(model_final, diabetes_test)

# evaluation metrics
params_model_final <- confusionMatrix_params(data = predicted_outcomes,
    reference = diabetes_test$outcome)
```

```r
knitr::kable(params_model_final, digits = 5)
```

f_value	accuracy	kappa	sensitivity	specificity	precision
0.84956	0.77922	0.45481	0.96	0.44444	0.7619

The final model achieves a F_1-score of 0.84956. Predictably, the final model performs worse on the `diabetes_test` set than on the `diabetes_validation` set. Although we reach high sensitivity, the specificity is below 0.5. This inevitably results in a moderate kappa (0.45481). This can also be seen in the confusion matrix, which shows a relatively high amount of false negatives.

```r
knitr::kable(data.frame(cm$table))
```

Prediction	Reference	Freq
no	no	48
yes	no	2
no	yes	15
yes	yes	12

7 Conclusion

The objective of the project was to build a machine learning model to predict the diabetes diagnosis of Pima Indians. We have developed various approaches to solve this classification problem. The final model showed high sensitivity but very low specificity. Thus, it should be possible to develop a predicitive model which can generate better results. Some approaches that could be tested are:

- building a stacked model,
- implementing more advanced machine learning models,
- using different imputation techniques.

References

Bano G. (2013). Glucose homeostasis, obesity and diabetes. Best Practice & Research Clinical Obstetrics & Gynaecology 27 (5): 715–726.

Knowler W.C., Pettitt D.J., Savage P.J. & Bennett P.H. (1981). Diabetes incidence in pima indians: Contributions of obesity and parental diabetes. American journal of epidemiology 113 (2): 144–156.

Murray C., Vos T., Lozano R., Naghavi M., Flaxman A., Michaud C. & Ezzati M. (2013). Years lived with disability (YLDs) for 1160 sequelae of 289 diseases and injuries 1990–2010: A systematic analysis for the global burden of disease study 2010. Lancet 381 (9867): 2197–2223.

Smith J.W., Everhart J.E., Dickson W., Knowler W.C. & Johannes R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In: Proceedings of the annual symposium on computer application in medical care. American Medical Informatics Association, p. 261.

Van Buuren S. & Groothuis-Oudshoorn K. (2011). Mice: Multivariate imputation by chained equations in r. Journal of statistical software 45: 1–67.