Отчет о выполнении индивидуального задания

### Разработка предиктивной модели для определения вероятности заболевания диабетом

Целью является прогнозировать вероятность наличия диабета и распределение по ней наблюдаемых объектов по двум классам: 0 - отсутствие и 1 - наличие соответсвующего заболевания

Это может быть полезно медицинским работникам при выявлении пациентов, которые могут быть подвержены риску развития диабета, и при разработке индивидуальных планов лечения.

### Описание используемой модели

Для предсказания использована одно из семейств моделей GLM Существует несколько семейств моделей GLM в зависимости от состава переменной отклика. Для работы выбрана биномиальное семейство. Выбор основан на том, что Биномиальное семейство используется для бинарных переменных отклика (т.е. двух категорий) и предполагает биномиальное распределение. Иными словами, в работе используется логистическая регрессия, которая будет определять зависимую бинарную переменную наличия заболевания от независимых переменных. Независимыми переменными будут служить все остальные переменные в train\_data в качестве предикторов. Это может включать различные факторы, такие как возраст, индекс массы тела, уровень глюкозы и т.д.

## Описание набора данных

Набор данных взят с сайта kaggle. Ссылка для скачивания: <https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset/data>

Набор данных для прогнозирования диабета представляет собой набор медицинских и демографических данных о пациентах, а также их диабетическом статусе (положительном или отрицательном). Эти данные включают такие характеристики, как возраст, пол, индекс массы тела (ИМТ), наличие гипертонии, наличие заболевания сердца, история курения, уровень HbA1c и глюкозы в крови. Этот набор данных используется для построения модели машинного обучения и прогнозирования диабета у пациентов на основе их истории болезни и демографической информации.

Это может быть полезно медицинским работникам при выявлении пациентов, которые могут быть подвержены риску развития диабета, и при разработке индивидуальных планов лечения.

### Подробный разбор кода

## Разработка модели

# Подключение необходимых пакетов

# libraries  
library("caret")  
library("ggplot2")  
library("ROCR")  
set.seed(42)

## Чтение и предобработка данных

1. Считывание данных из файла Загрузим данные из файла CSV с именем “diabetes\_prediction\_dataset.csv” и сохраним их в переменной my\_df. Также посмотрим структуру данных.

my\_df <- read.csv("diabetes\_prediction\_dataset.csv")  
str(my\_df)

## 'data.frame': 100000 obs. of 9 variables:  
## $ gender : chr "Female" "Female" "Male" "Female" ...  
## $ age : num 80 54 28 36 76 20 44 79 42 32 ...  
## $ hypertension : int 0 0 0 0 1 0 0 0 0 0 ...  
## $ heart\_disease : int 1 0 0 0 1 0 0 0 0 0 ...  
## $ smoking\_history : chr "never" "No Info" "never" "current" ...  
## $ bmi : num 25.2 27.3 27.3 23.4 20.1 ...  
## $ HbA1c\_level : num 6.6 6.6 5.7 5 4.8 6.6 6.5 5.7 4.8 5 ...  
## $ blood\_glucose\_level: int 140 80 158 155 155 85 200 85 145 100 ...  
## $ diabetes : int 0 0 0 0 0 0 1 0 0 0 ...

1. Преобразование столбцов в факторы Некоторые столбцы измерены не в количественных шкалах, их стоит перевести в фактор, что позволит R правильно обрабатывать категориальные переменные.

my\_df[, c(1, 3, 4, 5, 9)] <- lapply(my\_df[, c(1, 3, 4, 5, 9)], as.factor)  
str(my\_df)

## 'data.frame': 100000 obs. of 9 variables:  
## $ gender : Factor w/ 3 levels "Female","Male",..: 1 1 2 1 2 1 1 1 2 1 ...  
## $ age : num 80 54 28 36 76 20 44 79 42 32 ...  
## $ hypertension : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...  
## $ heart\_disease : Factor w/ 2 levels "0","1": 2 1 1 1 2 1 1 1 1 1 ...  
## $ smoking\_history : Factor w/ 6 levels "current","ever",..: 4 5 4 1 1 4 4 5 4 4 ...  
## $ bmi : num 25.2 27.3 27.3 23.4 20.1 ...  
## $ HbA1c\_level : num 6.6 6.6 5.7 5 4.8 6.6 6.5 5.7 4.8 5 ...  
## $ blood\_glucose\_level: int 140 80 158 155 155 85 200 85 145 100 ...  
## $ diabetes : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 1 1 ...

1. Удаление дубликатов и проверка на наличие пропущенных значений.

# удалим повторяющиеся записи  
my\_df <- unique(my\_df)  
# проверим на пропущенные значения  
any(is.na(my\_df))

## [1] FALSE

Пропущенных значений нет, теперь можно посмотреть краткую сводку о данных и перейти к обработке данных.

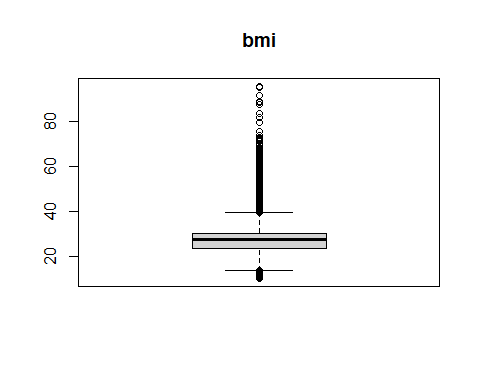
summary(my\_df)

## gender age hypertension heart\_disease smoking\_history   
## Female:56161 Min. : 0.08 0:88685 0:92223 current : 9197   
## Male :39967 1st Qu.:24.00 1: 7461 1: 3923 ever : 3998   
## Other : 18 Median :43.00 former : 9299   
## Mean :41.79 never :34398   
## 3rd Qu.:59.00 No Info :32887   
## Max. :80.00 not current: 6367   
## bmi HbA1c\_level blood\_glucose\_level diabetes   
## Min. :10.01 Min. :3.500 Min. : 80.0 0:87664   
## 1st Qu.:23.40 1st Qu.:4.800 1st Qu.:100.0 1: 8482   
## Median :27.32 Median :5.800 Median :140.0   
## Mean :27.32 Mean :5.533 Mean :138.2   
## 3rd Qu.:29.86 3rd Qu.:6.200 3rd Qu.:159.0   
## Max. :95.69 Max. :9.000 Max. :300.0

1. Удаление выбросов методом межквартильного размаха.

Есть подозрения на наличие выбросов в переменной bmi. Это можно проверить с помощью графика boxplot (ящик с усами)

boxplot(my\_df$bmi, main = "bmi")



Как видно, существует множество значений, выходящих за границу крайних квартелей. Удалим наблюдения с ними для лучшей работы моделей

# удаляем выбросы  
q\_1 <- quantile(my\_df$bmi, 0.25)  
q\_3 <- quantile(my\_df$bmi, 0.75)  
iqr <- q\_3 - q\_1  
lower\_bound <- q\_1 - 1.5 \* iqr  
upper\_bound <- q\_3 + 1.5 \* iqr  
my\_df <- my\_df[my\_df$bmi >= lower\_bound & my\_df$bmi <= upper\_bound, ]

1. Стандартизация данных.

my\_df[, -c(1, 3, 4, 5, 9)] <- sapply(my\_df[, -c(1, 3, 4, 5, 9)], scale)  
head(my\_df)

## gender age hypertension heart\_disease smoking\_history bmi  
## 1 Female 1.6912937 0 1 never -0.2332038  
## 2 Female 0.5442608 0 0 No Info 0.1737743  
## 3 Male -0.6027722 0 0 never 0.1737743  
## 4 Female -0.2498389 0 0 current -0.5656648  
## 5 Male 1.5148271 1 1 current -1.1981050  
## 6 Female -0.9557054 0 0 never 0.1737743  
## HbA1c\_level blood\_glucose\_level diabetes  
## 1 1.0133688 0.05707638 0  
## 2 1.0133688 -1.43018178 0  
## 3 0.1687918 0.50325382 0  
## 4 -0.4881014 0.42889092 0  
## 5 -0.6757852 0.42889092 0  
## 6 1.0133688 -1.30624360 0

1. Разделение данных на обучающую и тестовую выборки. В этом шаге данные разделяются на обучающую (70%) и тестовую (30%) выборки. Функция createDataPartition используется для случайного выбора индексов для обучающей выборки, а оставшиеся данные используются для тестовой выборки.

# split data  
train\_index <- createDataPartition(my\_df$diabetes, p = 0.7, list = FALSE)  
train\_data <- my\_df[train\_index, ]  
test\_data <- my\_df[-train\_index, ]

## Обучение модели и предсказание модели.

# train model on split data  
model <- glm(diabetes ~ ., data = train\_data, family = "binomial")  
  
predictions <- predict(model, newdata = test\_data[, c(-9)], type = "response")

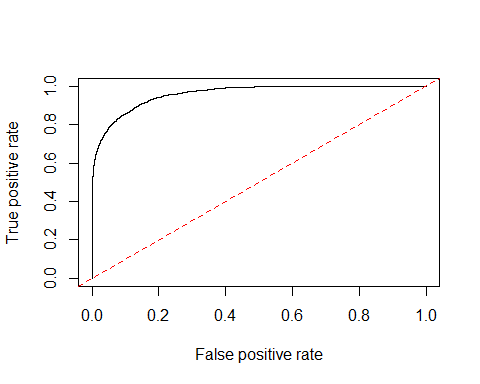
Выполнив код мы получили вероятности того, относится ли наблюдаемый объект к классу 1 (имеется диабет). Установив пороговое значение, мы сможем определять сам класс.

## Определение порогового значения

Теперь остро встает вопрос определения порогового значения для отнесения объекта к классу. Для этого в своих воспользуемся визуализацией метрик модели

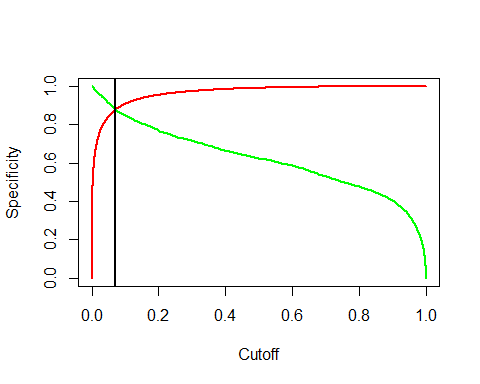
Для начала отобразим ROC curve, чтобы убедиться в хорошей точности модели и отсутствии перекошенности.

#####  
pred\_fit <- prediction(as.numeric(predictions), as.numeric(test\_data$diabetes))  
perf\_fit <- performance(pred\_fit, "tpr", "fpr")  
plot(perf\_fit)  
# Add a diagonal reference line  
abline(a = 0, b = 1, col = "red", lty = 2)



Теперь отобразим две метрики spec and sens. Их пересечение позволит определить оптимальнейшую границу.

#####  
perf3 <- performance(pred\_fit, x.measure = "cutoff", measure = "spec")  
perf4 <- performance(pred\_fit, x.measure = "cutoff", measure = "sens")  
  
plot(perf3, col = "red", lwd = 2)  
plot(add = T, perf4, col = "green", lwd = 2)  
# Отобразим линию пересечения на координате 0.07  
abline(v = 0.07, lwd = 2)



Так мы определили пороговое значение, и теперь можем поменять предсказанные вероятности на определенный класс.

threshold <- 0.07

# Определим классы и вектор с метками, которые говрят о правильности определения класса.

predicted\_classes <- ifelse(predictions > threshold, 1, 0)  
  
correct <- ifelse(predicted\_classes == test\_data$diabetes, 1, 0)

# Используем различные метрики точности модели.

predicted\_classes <- factor(predicted\_classes)  
test\_data$diabetes <- factor(test\_data$diabetes)  
  
confusion\_matrix <- confusionMatrix(predicted\_classes, test\_data$diabetes)  
print(confusion\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 21980 252  
## 1 3093 1912  
##   
## Accuracy : 0.8772   
## 95% CI : (0.8732, 0.8811)  
## No Information Rate : 0.9205   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4752   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8766   
## Specificity : 0.8835   
## Pos Pred Value : 0.9887   
## Neg Pred Value : 0.3820   
## Prevalence : 0.9205   
## Detection Rate : 0.8070   
## Detection Prevalence : 0.8162   
## Balanced Accuracy : 0.8801   
##   
## 'Positive' Class : 0   
##

#####  
  
accuracy <- mean(correct)  
f1\_score <- confusion\_matrix$byClass["F1"]

Итак, мы получили Confusion matrix (матрицу ошибок) с различными метриками и две основные: accuracy = 0.8771891 and f1 score = 0.9292887.

# Сохранение модели.

Сохраним модель вместе с пороговым значением в model.RData.

threshold <- 0.07  
save(model, threshold, file = "model.RData")