Research log

Dennis Scheper - 373689

2020-11-16

Table of content

1 Original dataset

1.1 Introduction

The article, "Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records" states that hyperglycemia management has a significant impact on the outcome and readmission rates hospitalized patients. The authors based this conclusion on the comprehensive assessment of 70.000 diabetes patient records retrieved from 140 US hospitals. The results that depict the relationship between readmission rates and HbA1c levels can further enlargement already existing diabetes tactics to reduce readmission rates.[1]

1.2 Dataset and Attributes Information

All information used in this article comes from a database consisting of 41 tables and totals 117 features, such as demographics (like gender, race, and age), inpatient or outpatient, and (in-hospital) mortality. Data came from 130 hospitals in the USA for over ten years (1998-2008) and contained around 74 million unique visits by 18 million unique patients. This research used information that needs to accede to the following specifications:

1. Is a hospital admission;
2. The encounter is a diagnosed with 'diabetes', any kind will satisfy;
3. The length of admission was at least one day up to eighteen days;
4. Laboratory test results are available; and
5. Medications were administered.

Of these five criteria points, 101.000 encounters fulfilled all specifications. After some considerations with removing encounters based on incomplete (weight and medical specialty) or biased data (discharge to a hospice or death), 69.984 encounters remained in the final dataset.

The initial dataset consists of 55 attributes, with the class attribute being an encounter of one patient. As there are way too many attributes to describe, please refer to the codebook for all descriptions; we only look at some essential attributes and their type and possible valuations. The age of a patient is nominal and is grouped into ten-year intervals. The admission type or the specific reason a patient was hospitalized came in 9 distinct values is nominal. Some attributes are numeric and count, such as the number of lab procedures, the number of medications, and the number of emergency visits. The database consists of three diagnosis attributes, which can have 848, 923, and 954 distinct values, respectively. The values are based on ICD9 three-letter codes and are of the nominal type. Other vital attributes are whether a patient changed medications (with the values 'no change' and 'change'; nominal type) or had diabetes medication ('yes' or 'no' values and nominal typing). Twenty-four

other attributes depict whether medicine is prescribed or not, if prescribed, then if the dosage was increased ('up'), decreased ('down'), or stayed the same ('steady') during the encounter. Readmission rates were calculated by looking at a nominal type ('Readmitted') with the possible valuations of '<30' for a patient that was readmitted within 30 days, '>30' for a patient that was readmitted after 30 days, and 'No' for patients that were not readmitted. The authors' goal was to determine whether a relationship between readmission rates and HbA1c measurement exists; therefore, they introduced a new attribute 'HbA1c' with four different valuations, based on the information from the database: 1) no HbA1c test performed; 2) HbA1c performed and in the normal range; 3) HbA1c performed and the result is greater than 8% with no change in diabetic medication; and 4) Hb1Ac performed, the result is more significant than 8%, and diabetic medication was changed.

1.3 Research Goal

Is it possible, using machine learning techniques to predict the duration of an inpatient hospital visit?

2 EDA (Exploratory Data Analysis)

This section will perform an exploratory data analysis (EDA) on the dataset described above. We can explore our dataset with an EDA and determine if any correlations exist between attributes and undermine any missing values. If any exist, we deal with them accordingly, looking to repair these values or remove them entirely. Additionally, we will look at variations between and in attributes for determining which ones are of most importance and interest to our research goals.

First, we load in our used packages, mostly used to visualize data and the data itself. Besides that, we also load in a codebook, which contains, i.e., a description for every attribute for the initial dataset. The codebook was not retrieved from any outside source but was constructed on the interpretation of the data.

```
# Load used packages
library(plyr)
library(dplyr, warn.conflicts = FALSE)
library(tidyr)
library(tidyverse)
library(ggplot2)
library(grid)
library(gridExtra)
library(hexbin)
library(foreign)
library(kableExtra)

# Load in used data
encounter.data <- read.table(
  file ='datasets/data_diabetes_retrieved/diabetic_data.csv',
  sep = ',', header = TRUE)

codebook <- read.csv(file = 'misc/codebook.csv', sep = ';',
                     header = T)
```

```r
# Read codebook via kbl
kable(codebook, "latex",
      caption = "Codebook with information about attributes",
      booktabs = T, longtable = T) %>%
 kable_styling(latex_options = c("repeat_header"),
               repeat_header_continued =
                 "\\textit{(Continued on Next Page...)}") %>%
  column_spec(1, width = "4cm") %>%
  column_spec(2, width = "3cm") %>%
  column_spec(3, width = "3cm") %>%
  column_spec(4, width = "5cm")
```

Table 1: Codebook with information about attributes

Name	Full.name	Data.type	Description.and.valuation
encounter_id	Encounter ID	Int	An unique number to identify an encounter.
patient_nbr	Patient number	Int	An unique number to identify a patient.
race	Race	Factor	The race of a patient with values: Caucasian, Asian, African American, Hispanic, and other.
gender	Gender	Factor	The gender of a patient with values: male, female, and unknown/invalid.
age	Age	Factor	The age of a patient, grouped in ten-year intervals, for example: [0, 10) and [90, 100).
weight	Weight	Factor	Weight in pounds.
adminsion_type_id	Admission type	Int	An integer identifying what a patient's admission type is, which corresponds with a method, for example:1 - emergency and 2 - urgent. Valuation has nine distinct values.

(Continued on Next Page...)

Table 1: Codebook with information about attributes (continued)

Name	Full.name	Data.type	Description.and.valuation
discharge_disposition_id	Discharge disposition	Int	An integer identifying how a patient was discharged, which corresponds with a method, for example: 1 - discharged to home and 2 - expired. Valuation has 29 distinct values.
admission_source_id	Admission source	Int	An integer identifying how a patient was admitted to hospital, which corresponds with a method, for example: 1 - physician referral and 2 - emergency room. Valuation has 21 distinct values.
time_in_hospital	Time in hospital	Int	Integer number of days between admission and discharge.
payer_code	Payer code	Factor	An integer identifying how a patient is paying, which corresponds with a method, for example: 1 - Blue Cross/Blue Shield and 2 - Medicare. Valuatoon has 23 distinct values.
medical_specialty	Medical specialty	Factor	An integer identifying the specialty of the admitting physician, which corresponds with a method, for example: 1 - cardiology and 2 - internal medicine. Valuation has 84 distinct values.
num_lab_procedures	Number of lab procedures	Int	Number of lab tests performed during the encounter.

(Continued on Next Page...)

Table 1: Codebook with information about attributes (continued)

Name	Full.name	Data.type	Description.and.valuation
num_procedures	Number of procedures	Int	Number of procedures, other than lab tests, performed during the encounter.
num_medications	Number of medications	Int	Number of distinct generic names administered during the encounter.
number_outpatient	Number of outpatient visits	Int	Number of outpatient visits of the patient in the year preceding the encounter.
number_emergency	Number of emergency visits	Int	Number of emergency visits of the patient in the year preceding the encounter.
number_inpatient	Number of inpatient visits	Int	Number of inpatient visits of the patient in the year preceding the encounter.
diag_1	Diagnosis 1	Factor	The primary diagnosis and is coded as first three digits of ICD9, with 848 distinct values.
diag_2	Diagnosis 2	Factor	The secondary diagnosis and is coded as first three digits of ICD9, with 923 distinct values.
diag_3	Diagnosis 3	Factor	An additional secondary diagnosis and is coded as first three digits of ICD9, with 954 distinct values.
number_diagnoses	Number of diagnoses	Int	Number of diagnoses entered to the system.
max_glu_serum	Glucose serum test results	Factor	Indicates the range of the result or if the test was not taken. Values: >200, >300 and normal if the test is taken, and none if the test is not taken.

(Continued on Next Page...)

Table 1: Codebook with information about attributes (continued)

Name	Full.name	Data.type	Description.and.valuation
A1Cresult	Alc test results	Factor	Indicates the range of the result or if the test was not taken. Values: >8% if result was greater than 8%, 7% if result was greater than 7% but less than 8%, normal if the test was less than 7%, and none if no test was taken.
metformin	Metformin prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.
repaglinide	Repaglinide prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.

(Continued on Next Page...)

Table 1: Codebook with information about attributes (continued)

Name	Full.name	Data.type	Description.and.valuation
nateglinide	Nateglinide prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.
chlorpropamide	Chlorpropamide prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.
glimepiride	Glimepiride prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.

(Continued on Next Page...)

Table 1: Codebook with information about attributes (continued)

Name	Full.name	Data.type	Description.and.valuation
acetohexamide	Acetohexamide prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.
glipizide	Glipizide prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.
glyburide	Glyburide prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.

(Continued on Next Page...)

Table 1: Codebook with information about attributes (continued)

Name	Full.name	Data.type	Description.and.valuation
tolbutamide	Tolbutamide prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.
pioglitazone	Pioglitazone prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.
rosiglitazone	Rosiglitazone prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.

(Continued on Next Page...)

Table 1: Codebook with information about attributes (continued)

Name	Full.name	Data.type	Description.and.valuation
acarbose	Acarbose prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.
miglitol	Miglitol prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.
troglitazone	Troglitazone prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.

(Continued on Next Page...)

Table 1: Codebook with information about attributes (continued)

Name	Full.name	Data.type	Description.and.valuation
tolazamide	Tolazamide prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.
examide	Examide prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.
citoglipton	Citoglipton prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.

(Continued on Next Page...)

Table 1: Codebook with information about attributes (continued)

Name	Full.name	Data.type	Description.and.valuation
insulin	Insulin prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.
glyburide-metformin	Glyburide-metformin prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.
glipizide-metformin	Glipizide-metaformin prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.

(Continued on Next Page...)

Table 1: Codebook with information about attributes (continued)

Name	Full.name	Data.type	Description.and.valuation
glimepiride-pioglitazone	Glimepiride-pioglitazone prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.
metformin-pioglitazone	Metformin-pioglitazone prescribed	Factor	Indicates whether the drug was prescribed or there was a change in dosage. Values: up if the dosage was increased, down if the dosage was decreased, steady if there was no change in dosage, and No if the drug was not prescribed.
change	Change of medications	Factor	Indicates if medication was changed, with values: 'Change' or 'No change'.
diabetesMed	Diabetes medication	Factor	Indicates if diabetes medication was prescribed, with values: 'Yes' and 'No'.
readmitted	Readmitted	Factor	Days to inpatient readmission, with values: <30 if the patient was readmitted in less than 30 days, >30 if the patient was readmitted in more than 30 days, and 'NO' for no record of readmission.

To get a better picture of our dataset's distribution, we called upon the function glimpse(). We notice that the data contains 50 columns/ attributes. Additionally, a summary() function gives an outline of every attribute, showing each level with a count of its valuation.

```r
# Give a glimpse of how the data is distributed
glimpse(encounter.data)
```

```
## Rows: 101,766
## Columns: 50
## $ encounter_id             <int> 2278392, 149190, 64410, 500364, 16680, 357...
## $ patient_nbr              <int> 8222157, 55629189, 86047875, 82442376, 425...
## $ race                     <fct> Caucasian, Caucasian, AfricanAmerican, Cau...
## $ gender                   <fct> Female, Female, Female, Male, Male, Male, ...
## $ age                      <fct> [0-10), [10-20), [20-30), [30-40), [40-50)...
## $ weight                   <fct> ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ...
## $ admission_type_id        <int> 6, 1, 1, 1, 1, 2, 3, 1, 2, 3, 1, 2, 1, 1, ...
## $ discharge_disposition_id <int> 25, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 3, 6,...
## $ admission_source_id      <int> 1, 7, 7, 7, 7, 2, 2, 7, 4, 4, 7, 4, 7, 7, ...
## $ time_in_hospital         <int> 1, 3, 2, 2, 1, 3, 4, 5, 13, 12, 9, 7, 7, 1...
## $ payer_code               <fct> ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ...
## $ medical_specialty        <fct> Pediatrics-Endocrinology, ?, ?, ?, ?, ?, ?...
## $ num_lab_procedures       <int> 41, 59, 11, 44, 51, 31, 70, 73, 68, 33, 47...
## $ num_procedures           <int> 0, 0, 5, 1, 0, 6, 1, 0, 2, 3, 2, 0, 0, 1, ...
## $ num_medications          <int> 1, 18, 13, 16, 8, 16, 21, 12, 28, 18, 17, ...
## $ number_outpatient        <int> 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ number_emergency         <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ...
## $ number_inpatient         <int> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ diag_1                   <fct> 250.83, 276, 648, 8, 197, 414, 414, 428, 3...
## $ diag_2                   <fct> ?, 250.01, 250, 250.43, 157, 411, 411, 492...
## $ diag_3                   <fct> ?, 255, V27, 403, 250, 250, V45, 250, 38, ...
## $ number_diagnoses         <int> 1, 9, 6, 7, 5, 9, 7, 8, 8, 8, 9, 7, 8, 8, ...
## $ max_glu_serum            <fct> None, None, None, None, None, None, None, ...
## $ A1Cresult                <fct> None, None, None, None, None, None, None, ...
## $ metformin                <fct> No, No, No, No, No, No, Steady, No, No, No...
## $ repaglinide              <fct> No, No, No, No, No, No, No, No, No, No, No...
## $ nateglinide              <fct> No, No, No, No, No, No, No, No, No, No, No...
## $ chlorpropamide           <fct> No, No, No, No, No, No, No, No, No, No, No...
## $ glimepiride              <fct> No, No, No, No, No, No, Steady, No, No, No...
## $ acetohexamide            <fct> No, No, No, No, No, No, No, No, No, No, No...
## $ glipizide                <fct> No, No, Steady, No, Steady, No, No, No, St...
## $ glyburide                <fct> No, No, No, No, No, No, No, Steady, No, No...
## $ tolbutamide              <fct> No, No, No, No, No, No, No, No, No, No, No...
## $ pioglitazone             <fct> No, No, No, No, No, No, No, No, No, No, No...
## $ rosiglitazone            <fct> No, No, No, No, No, No, No, No, No, Steady...
## $ acarbose                 <fct> No, No, No, No, No, No, No, No, No, No, No...
## $ miglitol                 <fct> No, No, No, No, No, No, No, No, No, No, No...
## $ troglitazone             <fct> No, No, No, No, No, No, No, No, No, No, No...
```

```
## $ tolazamide               <fct> No, No, No, No, No, No, No, No, No, No, No...
## $ examide                  <fct> No, No, No, No, No, No, No, No, No, No, No...
## $ citoglipton              <fct> No, No, No, No, No, No, No, No, No, No, No...
## $ insulin                  <fct> No, Up, No, Up, Steady, Steady, Steady, No...
## $ glyburide.metformin      <fct> No, No, No, No, No, No, No, No, No, No, No...
## $ glipizide.metformin      <fct> No, No, No, No, No, No, No, No, No, No, No...
## $ glimepiride.pioglitazone <fct> No, No, No, No, No, No, No, No, No, No, No...
## $ metformin.rosiglitazone  <fct> No, No, No, No, No, No, No, No, No, No, No...
## $ metformin.pioglitazone   <fct> No, No, No, No, No, No, No, No, No, No, No...
## $ change                   <fct> No, Ch, No, Ch, Ch, No, Ch, No, Ch, Ch, No...
## $ diabetesMed              <fct> No, Yes, Yes, Yes, Yes, Yes, Yes, Yes, Yes...
## $ readmitted               <fct> NO, >30, NO, NO, NO, >30, NO, >30, NO, NO,...
```

```
# Give a quick summary of the data,
# and give information such as min and max value, median and mean
summary(encounter.data)
```

```
##   encounter_id        patient_nbr                    race
##  Min.   :    12522   Min.   :      135   ?                : 2273
##  1st Qu.: 84961194   1st Qu.: 23413221   AfricanAmerican:19210
##  Median :152388987   Median : 45505143   Asian            :  641
##  Mean   :165201646   Mean   : 54330401   Caucasian       :76099
##  3rd Qu.:230270888   3rd Qu.: 87545950   Hispanic         : 2037
##  Max.   :443867222   Max.   :189502619   Other            : 1506
##
##              gender           age              weight      admission_type_id
##  Female         :54708   [70-80):26068   ?        :98569   Min.   :1.000
##  Male           :47055   [60-70):22483   [75-100) : 1336   1st Qu.:1.000
##  Unknown/Invalid:    3   [50-60):17256   [50-75)  :  897   Median :1.000
##                          [80-90):17197   [100-125):  625   Mean   :2.024
##                          [40-50): 9685   [125-150):  145   3rd Qu.:3.000
##                          [30-40): 3775   [25-50)  :   97   Max.   :8.000
##                          (Other): 5302   (Other)  :   97
##  discharge_disposition_id admission_source_id time_in_hospital   payer_code
##  Min.   : 1.000           Min.   : 1.000      Min.   : 1.000   ?      :40256
##  1st Qu.: 1.000           1st Qu.: 1.000      1st Qu.: 2.000   MC     :32439
##  Median : 1.000           Median : 7.000      Median : 4.000   HM     : 6274
##  Mean   : 3.716           Mean   : 5.754      Mean   : 4.396   SP     : 5007
##  3rd Qu.: 4.000           3rd Qu.: 7.000      3rd Qu.: 6.000   BC     : 4655
##  Max.   :28.000           Max.   :25.000      Max.   :14.000   MD     : 3532
##                                                                (Other): 9603
##                medical_specialty num_lab_procedures num_procedures
##  ?                    :49949   Min.   :  1.0      Min.   :0.00
```

```
##  InternalMedicine      :14635   1st Qu.: 31.0      1st Qu.:0.00
##  Emergency/Trauma      : 7565   Median : 44.0      Median :1.00
##  Family/GeneralPractice: 7440   Mean   : 43.1      Mean   :1.34
##  Cardiology            : 5352   3rd Qu.: 57.0      3rd Qu.:2.00
##  Surgery-General       : 3099   Max.   :132.0      Max.   :6.00
##  (Other)               :13726
##  num_medications number_outpatient number_emergency  number_inpatient
##  Min.   : 1.00   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.:10.00   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median :15.00   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   :16.02   Mean   : 0.3694   Mean   : 0.1978   Mean   : 0.6356
##  3rd Qu.:20.00   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 1.0000
##  Max.   :81.00   Max.   :42.0000   Max.   :76.0000   Max.   :21.0000
##
##      diag_1          diag_2          diag_3      number_diagnoses max_glu_serum
##  428    : 6862   276    : 6752   250    :11555   Min.   : 1.000   >200: 1485
##  414    : 6581   428    : 6662   401    : 8289   1st Qu.: 6.000   >300: 1264
##  786    : 4016   250    : 6071   276    : 5175   Median : 8.000   None:96420
##  410    : 3614   427    : 5036   428    : 4577   Mean   : 7.423   Norm: 2597
##  486    : 3508   401    : 3736   427    : 3955   3rd Qu.: 9.000
##  427    : 2766   496    : 3305   414    : 3664   Max.   :16.000
##  (Other):74419   (Other):70204   (Other):64551
##  A1Cresult     metformin     repaglinide     nateglinide     chlorpropamide
##  >7  : 3812   Down  :  575   Down  :    45   Down  :    11   Down  :     1
##  >8  : 8216   No    :81778   No    :100227   No    :101063   No    :101680
##  None:84748   Steady:18346   Steady:  1384   Steady:   668   Steady:    79
##  Norm: 4990   Up    : 1067   Up    :   110   Up    :    24   Up    :     6
##
##
##
##  glimepiride    acetohexamide    glipizide      glyburide     tolbutamide
##  Down  :  194   No    :101765   Down  :  560   Down  :  564   No    :101743
##  No    :96575   Steady:     1   No    :89080   No    :91116   Steady:    23
##  Steady: 4670                   Steady:11356   Steady: 9274
##  Up    :  327                   Up    :  770   Up    :  812
##
##
##
##  pioglitazone   rosiglitazone    acarbose        miglitol      troglitazone
##  Down  :  118   Down  :   87   Down  :     3   Down  :     5   No    :101763
##  No    :94438   No    :95401   No    :101458   No    :101728   Steady:     3
##  Steady: 6976   Steady: 6100   Steady:   295   Steady:    31
##  Up    :  234   Up    :  178   Up    :    10   Up    :     2
```

```
##
##
##
##   tolazamide     examide     citoglipton   insulin      glyburide.metformin
##  No    :101727   No:101766   No:101766   Down  :12218   Down  :     6
##  Steady:    38                           No    :47383   No    :101060
##  Up    :     1                           Steady:30849   Steady:   692
##                                           Up    :11316   Up    :     8
##
##
##
##  glipizide.metformin glimepiride.pioglitazone metformin.rosiglitazone
##  No    :101753       No    :101765            No    :101764
##  Steady:    13       Steady:     1            Steady:     2
##
##
##
##
##
##  metformin.pioglitazone change     diabetesMed readmitted
##  No    :101765          Ch:47011   No :23403   <30:11357
##  Steady:     1          No:54755   Yes:78363   >30:35545
##                                                NO :54864
##
##
##
##
```

2.1 Missing values

As we see in the result determined by the glimpse() function, the columns race, weight, payer_code, gender, diag_3, and medical specialty have some or significant amounts of missing values. To get a better picture, we will zoom in on these attributes and determine the amount and the percentage of missing values.

```
myvars <- c('race', 'weight', 'payer_code',
            'medical_specialty', 'gender', 'diag_3')
amountRecords <- length(rownames(encounter.data))
tableMissingValues <- encounter.data %>%
  summarise_each_(funs("TotalMissing" =
                  sum(. %in% c('?', 'Unknown/Invalid'), na.rm = TRUE),
                       "MissingValuesPercentage" =
                 round(sum(. %in% c('?', 'Unknown/Invalid'),
                           na.rm = TRUE)/amountRecords * 100, 2)),
                  myvars) %>%
  t()
tableMissingValues <- data.frame("Missing.Values" =
                                   tableMissingValues[1:6,],
                                 "Missing.Values.Percentage" =
                                   tableMissingValues[7:12,])
rownames(tableMissingValues) <- myvars

kable(tableMissingValues, "latex", booktabs = T,
      caption = "Attributes that are missing values
      and their the amount of missing data" ) %>%
  kable_styling(full_width = F, latex_options = "hold_position") %>%
  column_spec(1, bold = T)
```

Table 2: Attributes that are missing values and their the amount of missing data

	Missing.Values	Missing.Values.Percentage
race	2273	2.23
weight	98569	96.86
payer_code	40256	39.56
medical_specialty	49949	49.08
gender	3	0.00
diag_3	1423	1.40

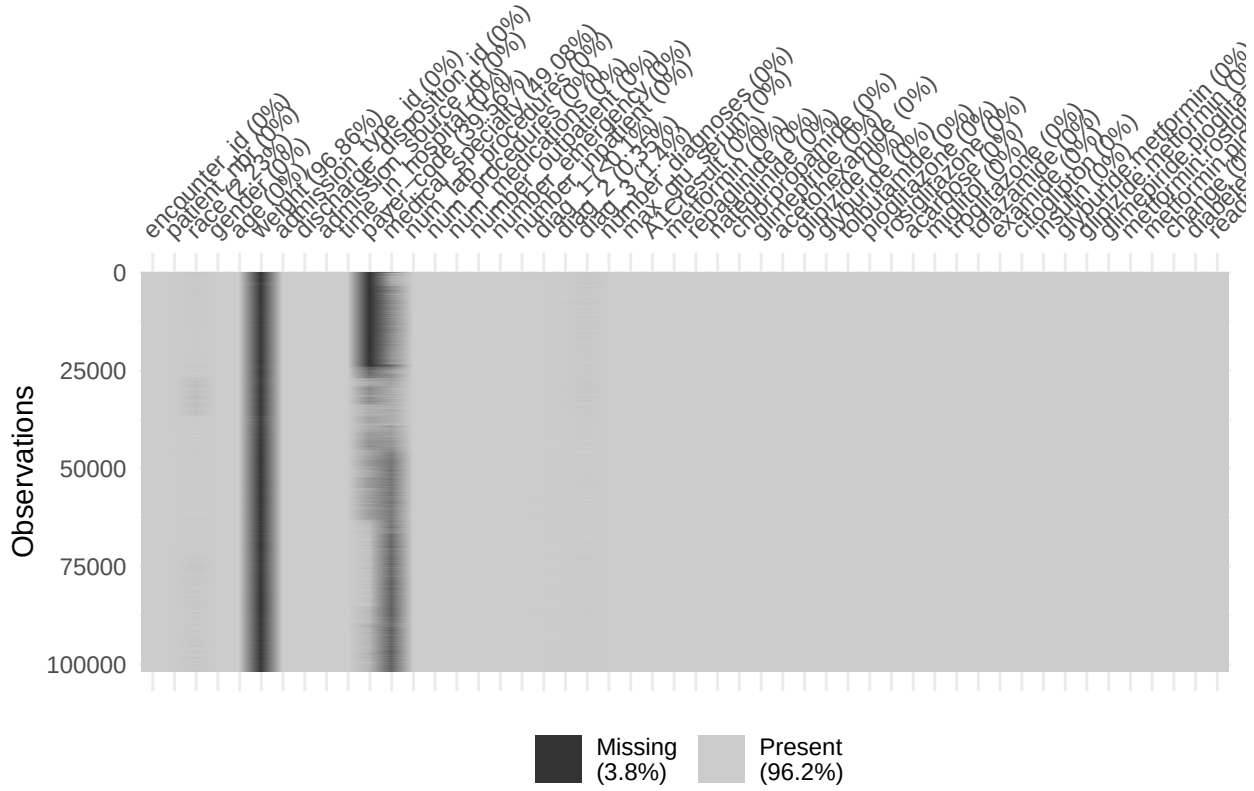


Figure 1: Missing values per attribute (depicted in black).

As we notice in table 2 and figure 1, attribute weight is almost entirely made-up of missing values. For that reason, it is a candidate for removal. The same can be said for payer_code and medical_specialty, where 39.56% and 49.08% of values are missing, respectively. Payer_code has no significant value to our research goals and questions; therefore, it can be removed. Medical_specialty can be of value as it contains a range of useful information. Missing values can be set to 'Missing' or reevaluated based on other valuations; doing this is not without risk, as it is almost half of the attributes' values. Deleting all missing values is not doable as it removes half of all records! Setting it to 'Missing' seems to be the most logical option. Attributes race and diag_3 seem to have little amounts of missing data, but removing them can alter our outcomes; therefore, we will use the same approach as medical_specialty. Gender only has three missing values. Considering the amount does not account for even one percent, removal does not seem to harm, and gender is, in most cases, considered binary data.

In the next section, we will introduce a new attribute (HbA1c measurement) based on an A1C test and the response to that result, defined as a change in diabetic medication. This test was performed at the time of hospital admission. We consider four groups of encounters:

1. no HbA1c test performed;
2. HbA1c performed and in normal range;
3. HbA1c performed and the result is greater than 8
4. HbA1c performed, result is greater than 8

```
encounter.data <- encounter.data %>%
  mutate(hba1c_res=ifelse((A1Cresult=='None'), 'one', NA)) %>%
  mutate(hba1c_res=ifelse((A1Cresult %in% c('Norm', '>7'))
                          & is.na(hba1c_res), 'two', hba1c_res)) %>%
  mutate(hba1c_res=ifelse((A1Cresult=='>8') & (change=='No')
                          & is.na(hba1c_res), 'three', hba1c_res)) %>%
  mutate(hba1c_res=ifelse((A1Cresult=='>8')
                          & (change=='Ch')
                          & is.na(hba1c_res), 'four', hba1c_res))
encounter.data$hba1c_res <- as.factor(encounter.data$hba1c_res)
```

We have many near-zero variance variables comprised in our dataset. For example, the attribute examide consists of entirely one level with no missing values. For that reason, it is a zero-variance attribute. This attribute is not informative and will not help in predicting an outcome. All other (near) zero-variance attributes were removed as they would increase computing times if kept. We are talking about the following eighteen attributes:

```
library(caret)
removal.names <- nearZeroVar(encounter.data, names = T)
encounter.data <- encounter.data %>%
  select(-c(removal.names))
```

Before we start analyzing our dataset, it is crucial to check on duplicates as our dataset contains multiple inpatient visits for some patients. These observations cannot be statistically independent and would create noise. We thus declare that only one encounter per patient is optimal.

```
duplicateCheck <- encounter.data %>%
  group_by(patient_nbr) %>%
  summarise(count = n()) %>%
  filter(count > 1)

table.dup <- data.frame('Initial number of records' = nrow(encounter.data),
```

```
                        'Number of Duplicates' = nrow(duplicateCheck),
                        'Difference in percent' = round((
                          nrow(encounter.data) - nrow(duplicateCheck) -
                          nrow(encounter.data)) /
                          nrow(encounter.data) * 100, 2) )

kbl(table.dup, booktabs = T,
    caption = "The effects of removing data") %>%
  kable_styling(full_width = F, latex_options = "hold_position") %>%
  column_spec(1, bold = T)
```

Table 3: The effects of removing data

Initial.number.of.records	Number.of.Duplicates	Difference.in.percent
101766	16773	-16.48

Looking at table 3, we can see that the initial dataset started with 101.766 records, of which 16.773 were duplicates. Potential removal of these records would leave us with 84993 records, a total loss of 16.48%—a small price to pay for a statistically independent dataset.

2.2 Categorial attributes

In this section, we take a look at variations in and between attributes. The attribute gender consists of three values ('female,' 'male,' and 'missing/unknown'). Since observe and use this attribute, it is essential to remove the abundant value.

```
# First, we need to remove all records containing the third option
# of gender: 'Missing/unknown', as this can be lethal in further analysis
encounter.data <- encounter.data %>%
  select(everything()) %>%
  filter(gender != 'Unknown/Invalid') %>%
  droplevels()

# Count specific variables of interest
agg <- count(encounter.data, age, gender, A1Cresult, race)

# Specify a color per value for visualization
ecols <- c(Female = "pink", Male = "blue2")
p1 <- ggplot(agg) +
       geom_col(aes(x = race, y = n, fill = gender)) +
       scale_fill_manual(values = ecols) +
       labs(x ="Race", y = "Total counts")
p2 <- ggplot(agg) +
       geom_col(aes(x = age, y = n, fill = gender)) +
       scale_fill_manual(values = ecols) +
       labs(x ="Age", y = "Total counts")
ecols <- c(">7" = "blue", ">8" = "orange",
           "None" = "green", "Norm" = "yellow")
p3 <- ggplot(agg) +
       geom_col(aes(x = age, y = n, fill = A1Cresult)) +
       scale_fill_manual(values = ecols) +
       labs(x ="Age", y = "Total counts")
```

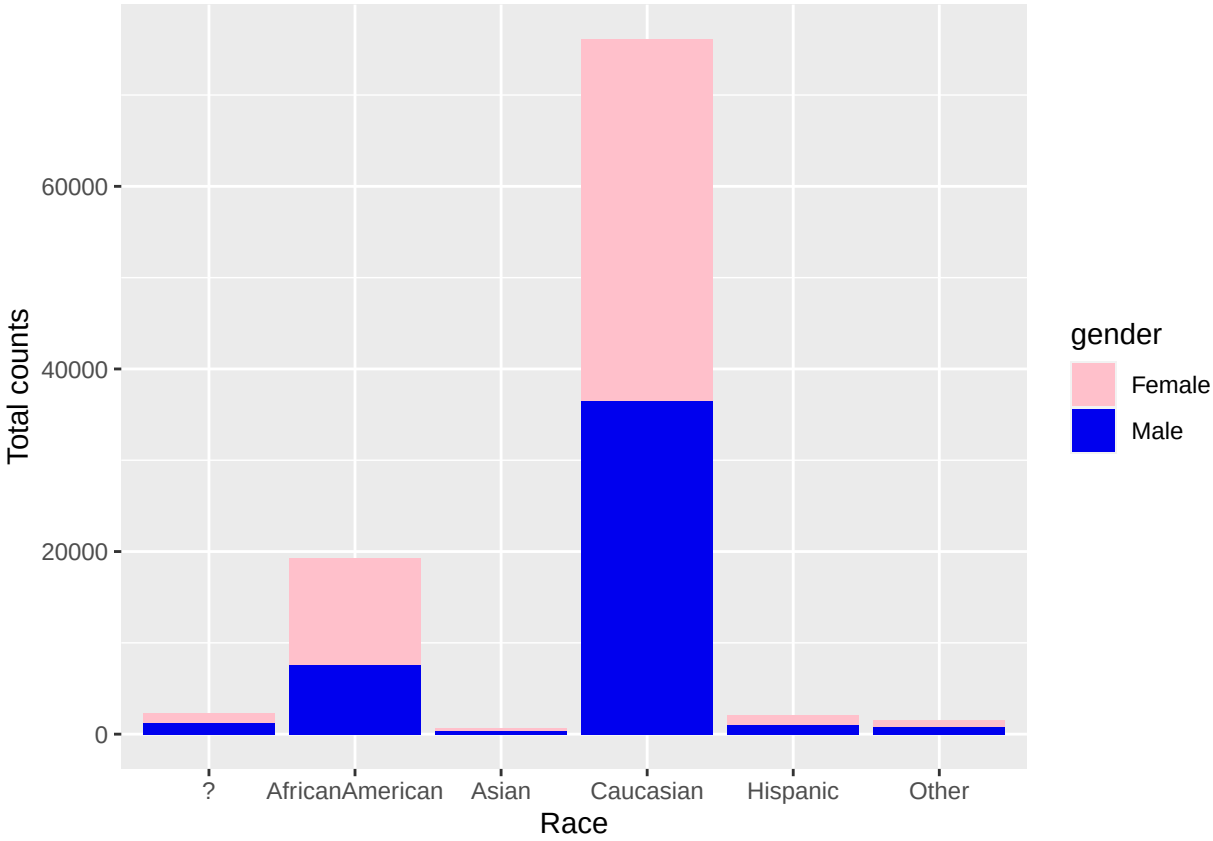


Figure 2: Race distributed with gender

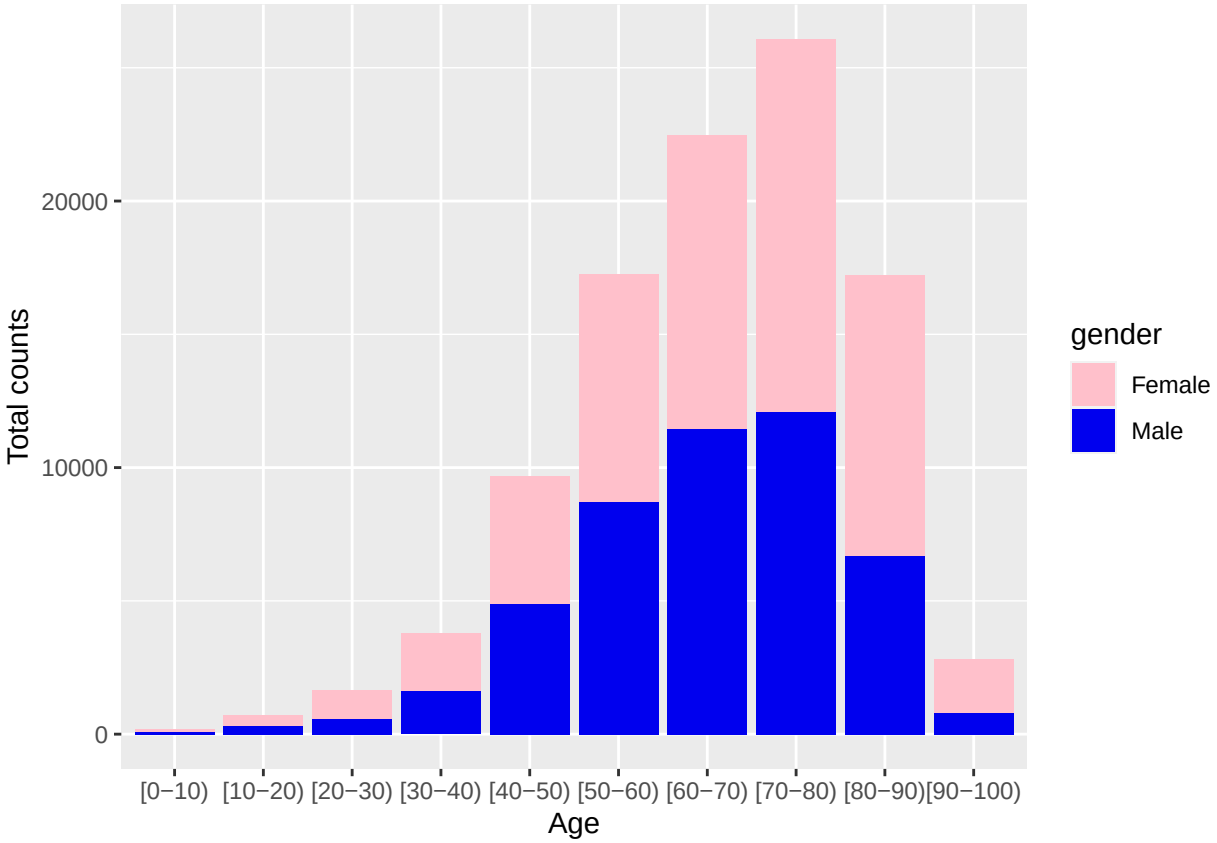


Figure 3: Age distributed with gender

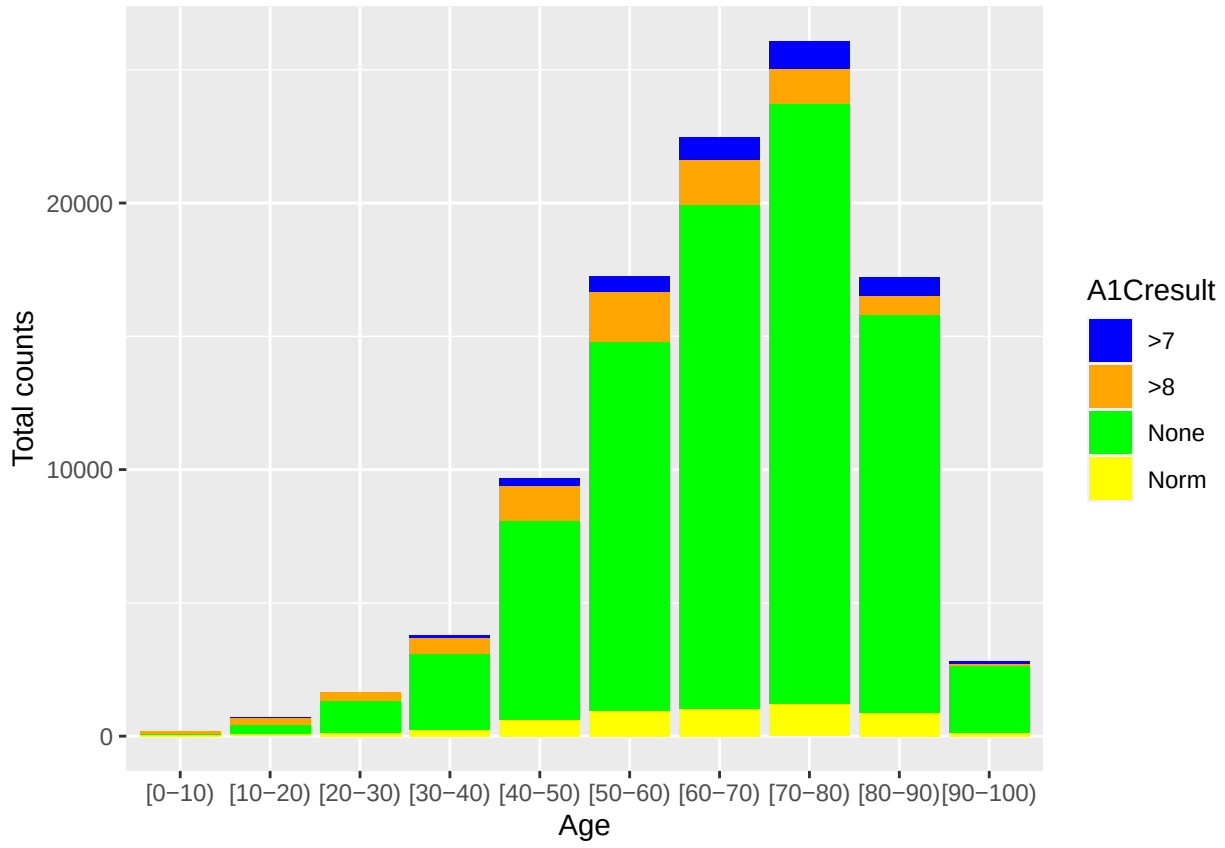


Figure 4: Age distributed with A1C test result

Looking at figure 2, we notice that most patients are from the Caucasian race with almost equivalent male and female ratios. We expect a rise in the number of diabetes patients when looking at older population groups. Figure 3 shows this exact prediction; the data is negatively skewed and increases in numbers per older age group. We observe the same results in figure 4, where an increase in the A1C test is shown when comparing older population groups. Additionally, the figure gives an essential insight for many patients - in most cases, the test was not conducted and shows that strategies surrounding testing diabetes are not normalized in hospital protocols.

This graph does not, however, make a distinction between non-ICU and ICU patients. The authors of the original dataset analyses stated that ICU departments' protocols have a stricter policy surrounding testing for diabetes. When comparing non-ICU and ICU patient records, we need to construct a new attribute called 'icu_or_non,' comprising data from attributes admission_type_id, admission_source_id, and discharge_disposition_id. We distinguish between non-ICU and ICU patients.

```
encounter.data <- encounter.data %>%
  # Removing dead patients
  filter(discharge_disposition_id != 11) %>%
  # Distinction between ICU and non-ICU patients
```

```
  mutate(admission_type_id =
           ifelse(admission_type_id %in% c(1, 2, 7),
                  'ICU patient', 'Non-ICU patient')) %>%
  mutate(admission_source_id =
           ifelse(admission_source_id %in% c(4, 7, 10, 12, 26),
                  'ICU patient', 'Non-ICU patient')) %>%
  mutate(discharge_disposition_id =
           ifelse(discharge_disposition_id %in% c(13, 14, 19, 20, 21),
                  'ICU patient', 'Non-ICU patient'))
# Now that we changed the labels, let us discover
# how many ICU and non-ICU patients we have
encounter.data <- encounter.data %>%
  # All columns need to be equal to each other
  mutate(icu_or_non = ifelse(admission_source_id == discharge_disposition_id
                             & admission_type_id == discharge_disposition_id,
                             admission_source_id,
                             ifelse(admission_source_id ==
                                    discharge_disposition_id,
                                    admission_source_id,
                                    admission_source_id)))
encounter.data$icu_or_non <- as.factor(encounter.data$icu_or_non)

agg <- count(encounter.data, gender, A1Cresult, icu_or_non, race)
ecols <- c(Female = "pink", Male = "blue2")
p1 <- ggplot(agg) +
       geom_col(aes(x = icu_or_non, y = n, fill = gender)) +
       scale_fill_manual(values = ecols) +
       labs(x ="ICU or non-ICU patient", y = "Total counts")

p2 <- ggplot(agg) +
       geom_col(aes(x = icu_or_non, y = n, fill = race)) +
       labs(x ="ICU or non-ICU patient", y = "Total counts")

ecols <- c('ICU patient' = 'blue', 'Non-ICU patient' = 'red')
p3 <- ggplot(agg) +
       geom_col(aes(x = A1Cresult, y = n, fill = icu_or_non)) +
       scale_fill_manual(values = ecols) +
       labs(x ="ICU or non-ICU patient", y = "Total counts")
```

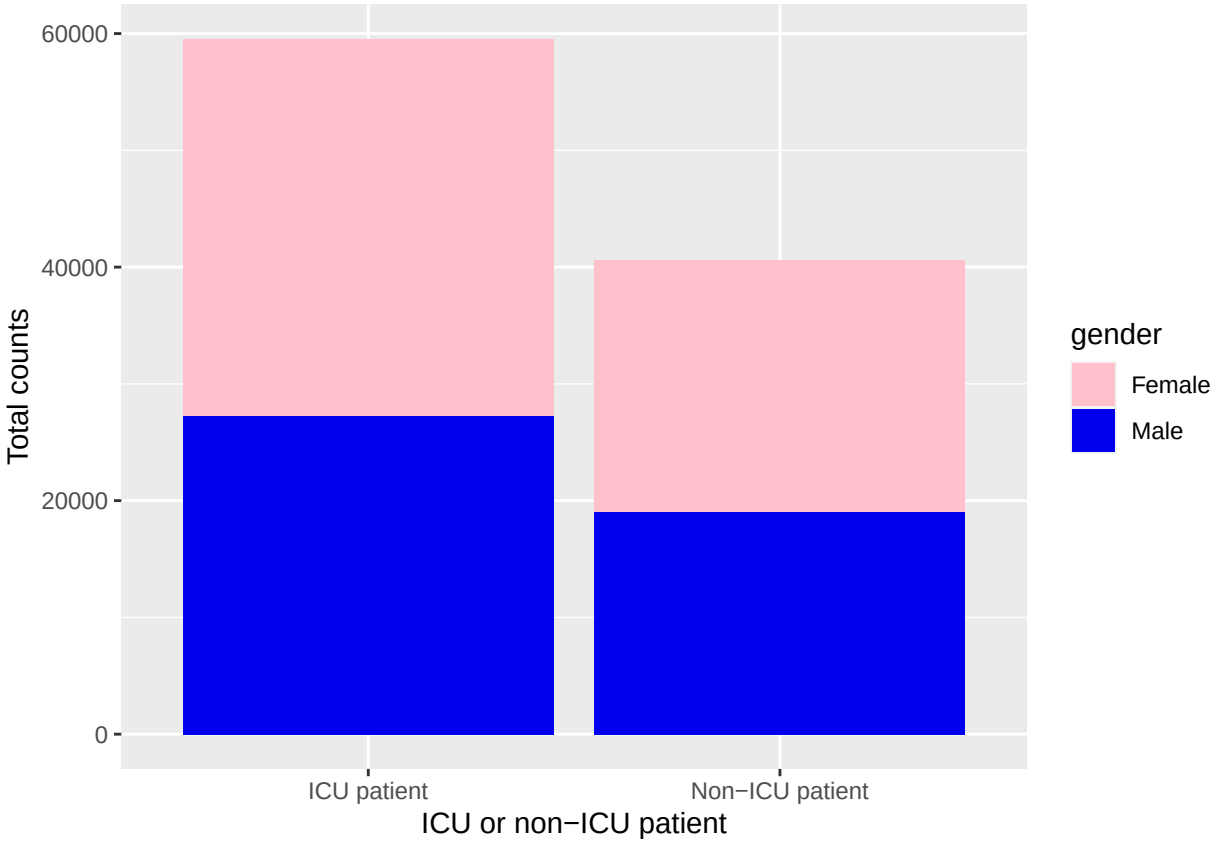


Figure 5: ICU statistics distributed with gender

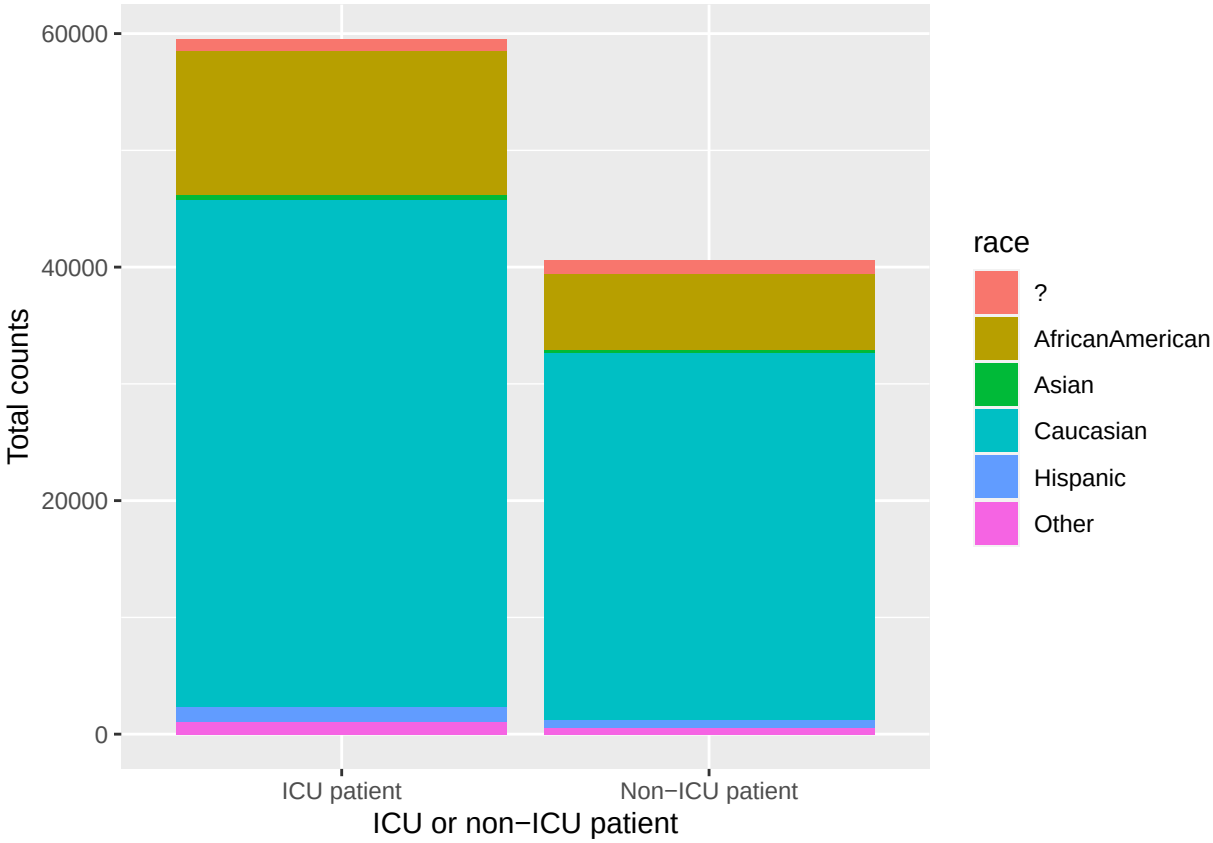


Figure 6: ICU statistics distributed with race

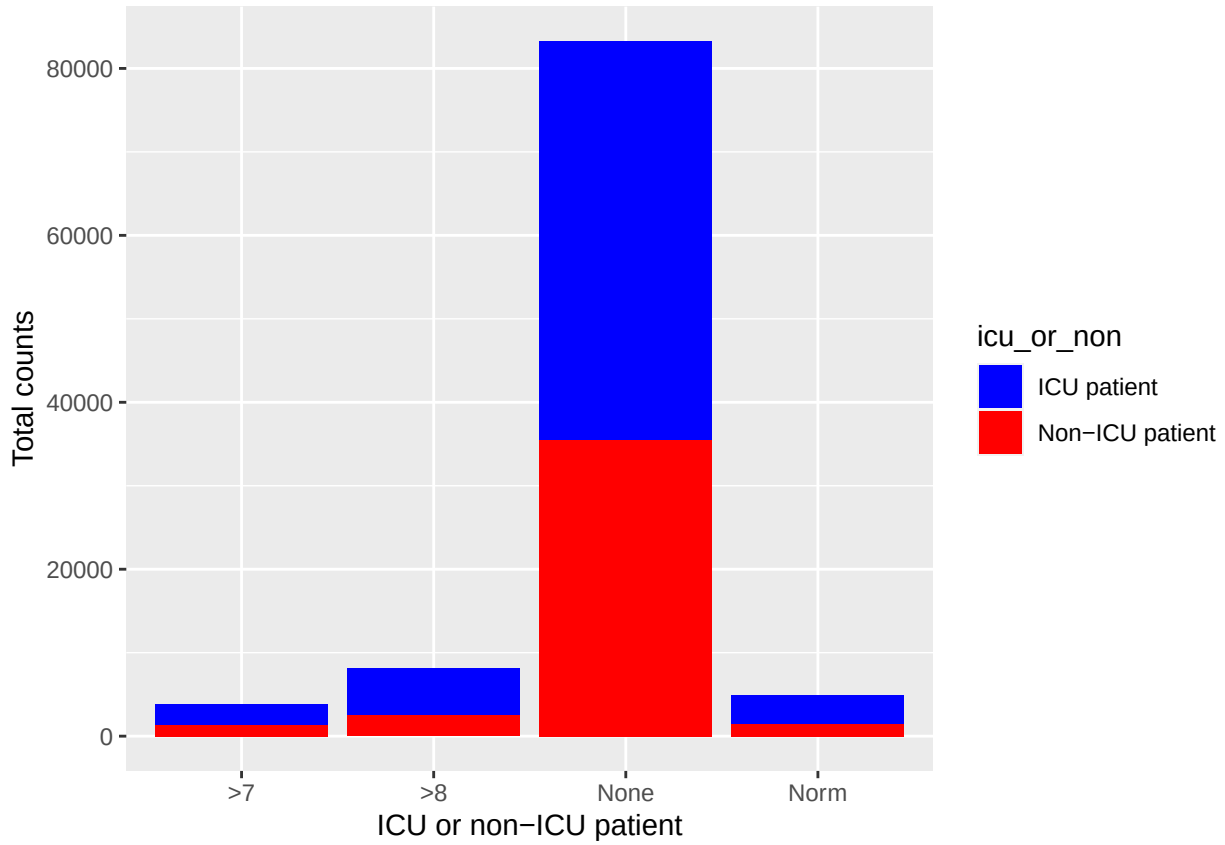


Figure 7: ICU statistics distributed with A1C test result

We observe the distinction between ICU and non-ICU in figures 5, 6, and 7. We notice a smaller population of non-ICU patients in all mentioned figures, whereas the ratio of not only gender but also seen in 6, where race is depicted, ratios stay consequently the same. Figure 7 shows that no matter the patient type (ICU or non-ICU), protocols surrounding diabetes testing is not firmly conducted. Keeping the distinction between non-ICU and ICU might not be necessary. However, it can be of complimentary use when starting with machine learning; the data is binary, and it allows the removal of three attributes.

The attributes diag_1, diag_2, and diag_3 consist of many three-digit ICD codes. Many of these codes belong together in a subgroup. In the following section, we will construct a better way of describing the exact diagnosis. ICD codes descriptions are retrieved from [2].

```
encounter.data <- encounter.data %>%
  mutate(diag_1 = case_when(diag_1 %in% c(390:459) ~ 'Circulatory',
                            diag_1 %in% c(460:519) ~ 'Respiratory',
                            diag_1 %in% c(520:579) ~ 'Digestive',
                            diag_1 %in% c(240:279) ~ 'Diabetes',
                            diag_1 %in% c(800:999) ~ 'Injury',
                            diag_1 %in% c(710:739) ~ 'Muscoloskeletal',
                            diag_1 %in% c(580:629) ~ 'Genitourinary',
```

```
                              TRUE ~ 'Others'))

encounter.data <- encounter.data %>%
  mutate(diag_2 = case_when(diag_2 %in% c(390:459) ~ 'Circulatory',
                              diag_2 %in% c(460:519) ~ 'Respiratory',
                              diag_2 %in% c(520:579) ~ 'Digestive',
                              diag_2 %in% c(240:279) ~ 'Diabetes',
                              diag_2 %in% c(800:999) ~ 'Injury',
                              diag_2 %in% c(710:739) ~ 'Muscoloskeletal',
                              diag_2 %in% c(580:629) ~ 'Genitourinary',
                              TRUE ~ 'Others'))
encounter.data <- encounter.data %>%
  mutate(diag_3 = case_when(diag_3 %in% c(390:459) ~ 'Circulatory',
                              diag_3 %in% c(460:519) ~ 'Respiratory',
                              diag_3 %in% c(520:579) ~ 'Digestive',
                              diag_3 %in% c(240:279) ~ 'Diabetes',
                              diag_3 %in% c(800:999) ~ 'Injury',
                              diag_3 %in% c(710:739) ~ 'Muscoloskeletal',
                              diag_3 %in% c(580:629) ~ 'Genitourinary',
                              TRUE ~ 'Others'))

# Convert to factors again
cols <- c('diag_1', 'diag_2', 'diag_3')
encounter.data[cols] <- lapply(encounter.data[cols], factor)
agg <- count(encounter.data, gender, diag_1, diag_2, diag_3)

# Specify a color per value for visualization
ecols <- c(Female = "pink", Male = "blue2")
p1 <- ggplot(agg) +
       geom_col(aes(x = diag_1, y = n, fill = gender)) +
       scale_fill_manual(values = ecols) +
      labs(x ="Primary diagnosis [category]", y = "Total counts")
p2 <- ggplot(agg) +
       geom_col(aes(x = diag_2, y = n, fill = gender)) +
       scale_fill_manual(values = ecols) +
       labs(x ="Secondary diagnosis [category]", y = "Total counts")
p3 <- ggplot(agg) +
       geom_col(aes(x = diag_3, y = n, fill = gender)) +
       scale_fill_manual(values = ecols) +
       labs(x ="Final diagnosis [category]", y = "Total counts")
```

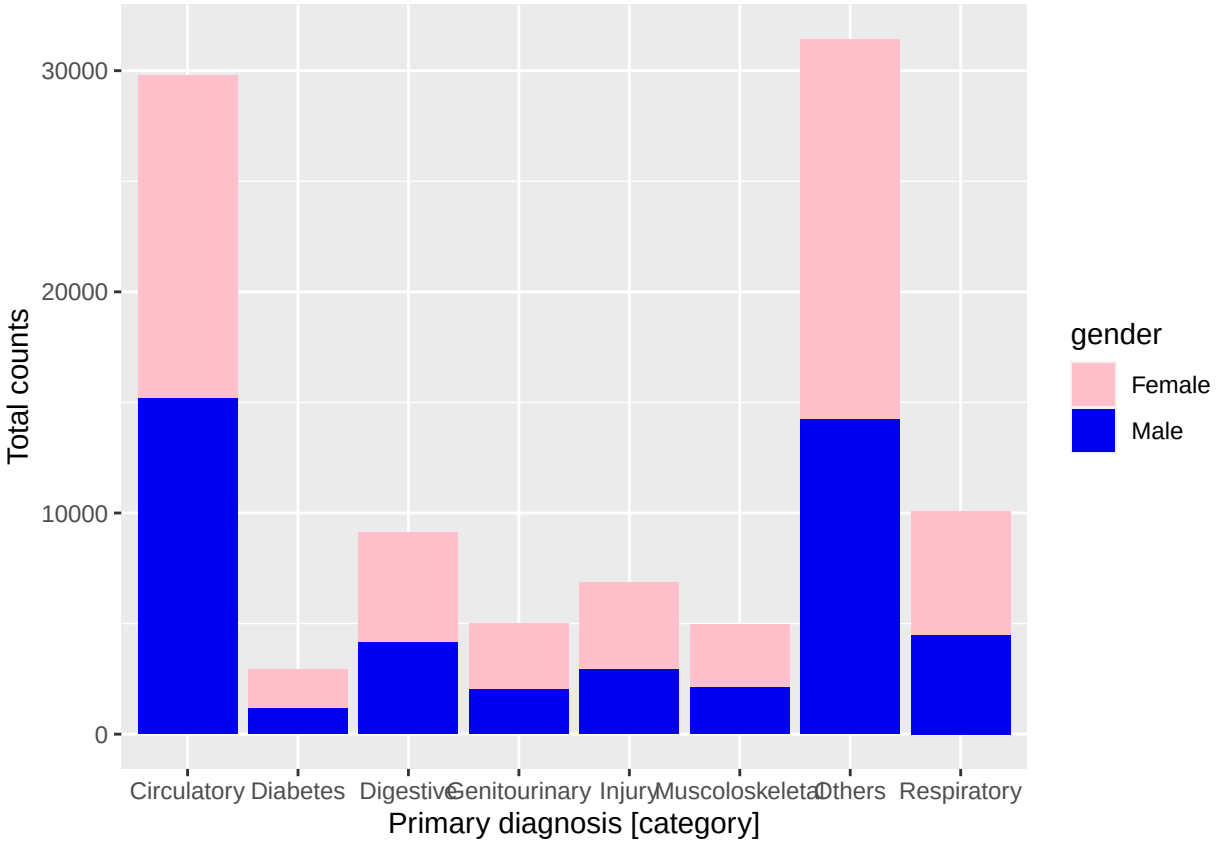


Figure 8: Primary diagnosis distributed with gender

Figure 9: Secondary diagnosis distributed with gender

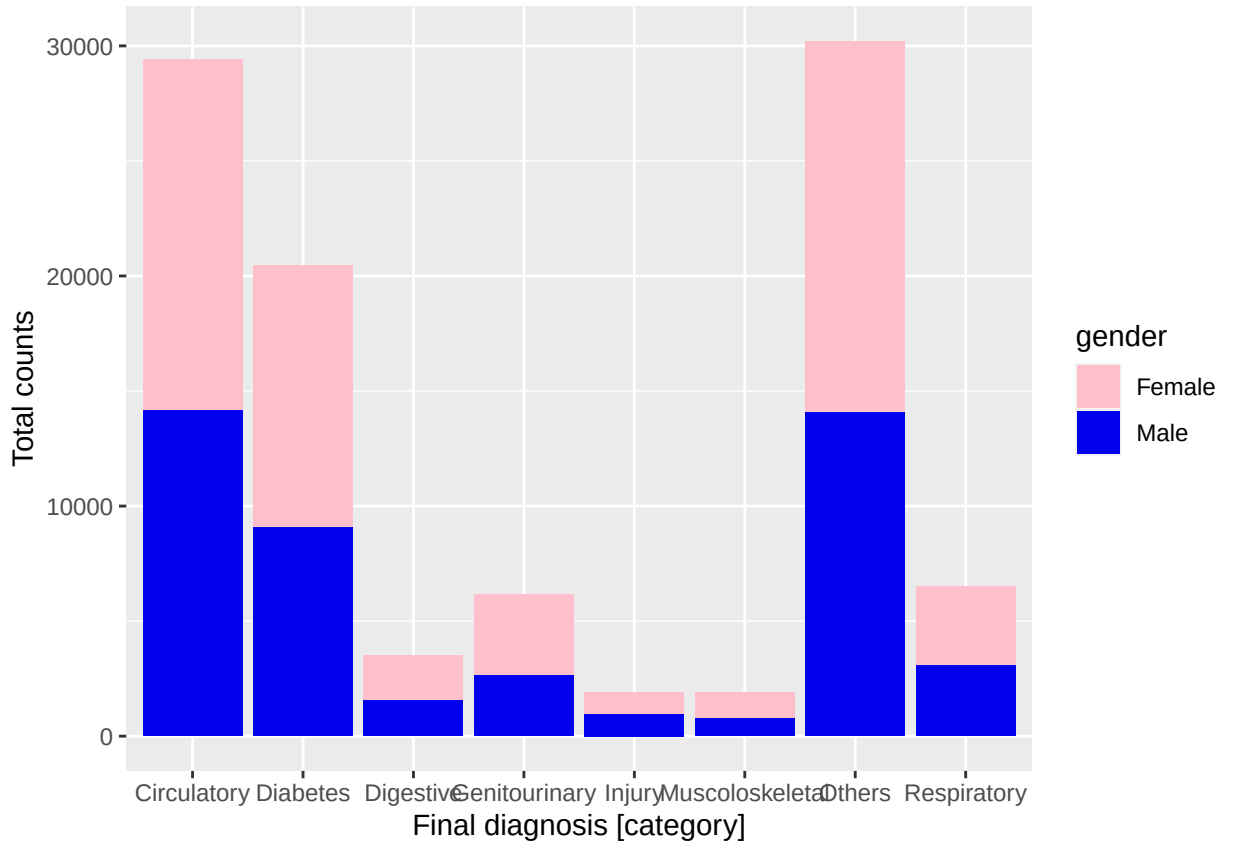


Figure 10: Final diagnosis distributed with gender

Looking at figure 8, 9, and 10, we can see the results of the revaluation of attributes diag_1, diag_2, and diag_3. These attributes depict the primary, secondary, and final diagnosis of a patient, respectively. Interestingly, the difference between the figures is the increase in the number of diabetes diagnoses. Due to not testing diabetes on the initial hospitalization, readmission rates increase as a patient's primary diagnosis is not sustainable. The data classification now shows a clearer picture and would undoubtedly be of fair use in the final dataset. The original authors did not keep diag_2 and diag_3, making records too complex to achieve their goals. Removal of these two attributes is still up for discussion, but the analysis does not give - for now - a clear indication for potential removal.

In the next section, we look at attribute medical_specialty and decide whether it is useful enough - considering the number of missing values - to be a candidate for the final dataset. We construct multiple plots that zoom in on the categories and valuations of this attribute.

```
library(pals)

# Count data we want to compare
agg <- count(encounter.data, age, medical_specialty, gender)

# Determine a color scheme for 71 values
```

```r
ecols <- c(alphabet(26), cols25(25), glasbey(22))

# Order variables from high to low via '-n'
agg_ord <- mutate(agg,
                  age = reorder(age, -n, sum),
                  medical_specialty = reorder(medical_specialty, -n, sum))
# p1: bar plot combined, and p2: per gender
p1 <- ggplot(agg_ord) +
      geom_col(aes(x = age, y = n, fill = medical_specialty)) +
      scale_fill_manual(values = ecols) +
      theme(legend.position = 'none') +
      labs(x ="Age [group]", y = "Total counts")

p1 <- p1 + facet_wrap(~gender) +
      theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
# Do the same as above, only now with pie charts
p2 <- ggplot(agg_ord) +
      geom_col(aes(x = 1, y = n, fill = medical_specialty),
               position = "fill") +
      coord_polar(theta = "y") +
      scale_fill_manual(values = ecols) +
      theme(legend.position = 'none')
p2 <- p2 + facet_wrap(~gender) +
    theme_bw() +
    theme(legend.position = 'none')
```

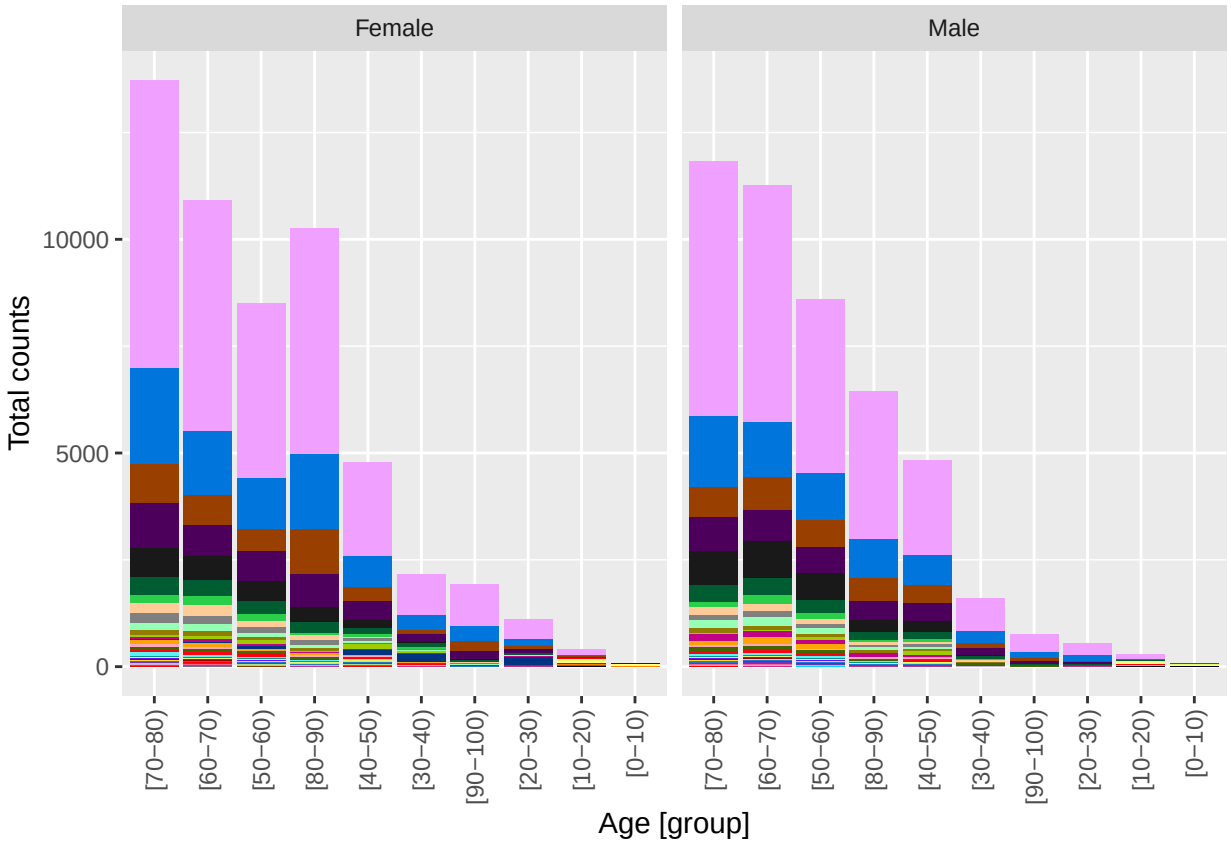


Figure 11: Medical specialty distributed with age groups and divided into gender

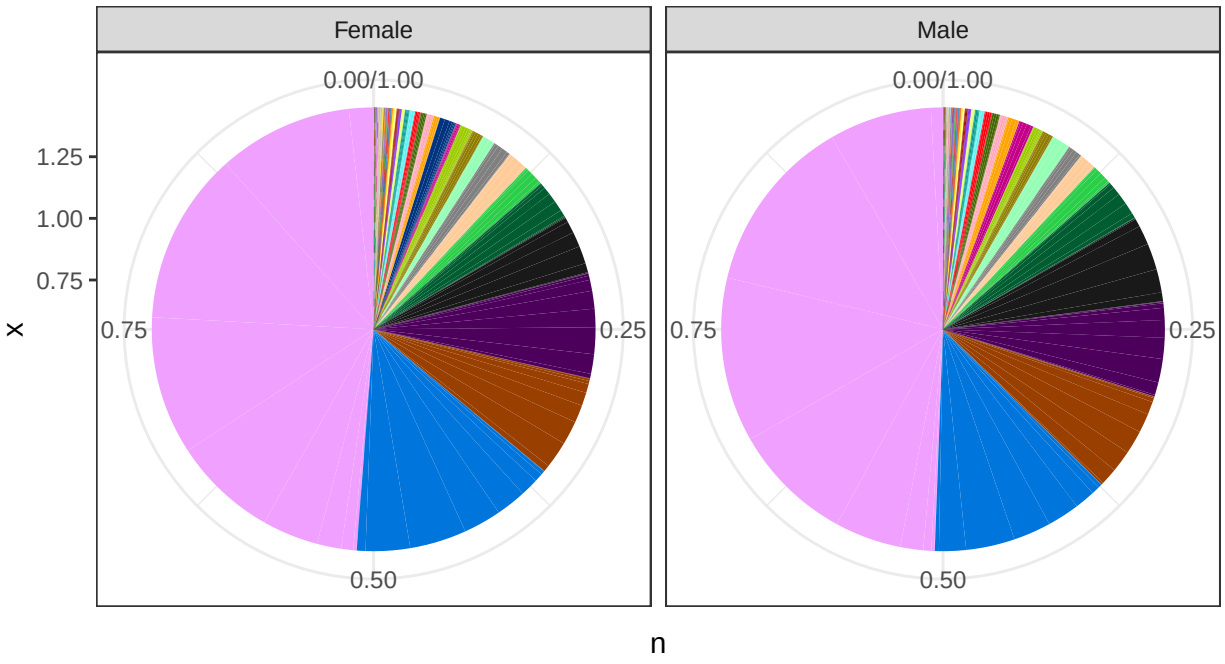


Figure 12: Medical specialty distributed with age groups and divived into gender

Looking at both figures 11 and 12, we observe a large valuation between all demographic attributes used. Depicted in purple are the missing values, as already established, make-up almost 50 percent of total records. Medical specialty is different from, for example, admission_id, where we could shrink the attribute to a better format. This is an alternative to this attribute. We are considering this because medical specialty does not give better information than any diagnosis attribute, which also gives, maybe even more useful, guidance about an encounter's medical history. At this stage, removal of this attribute could not be of lethal harm.

For our final categorical attribute, and one of the most influential -according to the original authors- we will discuss readmitted. Readmitted is an attribute with three different valuations: '<30' for readmission within 30 days after release, '>30' for readmission after 30 days release, and 'NO' for no readmission.

```
agg <- count(encounter.data, readmitted, race, gender, age, diag_1, diag_2)
ecols <- c('<30' = 'red', '>30' = 'blue', 'NO' = 'pink')

p1 <- ggplot(agg) +
       geom_col(aes(x = race, y = n, fill = readmitted)) +
       scale_fill_manual(values = ecols) +
```

```
       labs(x ="Race [group]", y = "Total counts")
p1 <- p1 + facet_wrap(~gender) +
       theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
p2 <- ggplot(agg) +
       geom_col(aes(x = age, y = n, fill = readmitted)) +
       scale_fill_manual(values = ecols) +
       labs(x ="Age [group]", y = "Total counts")
p2 <- p2 + facet_wrap(~gender) +
       theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
p3 <- ggplot(agg) +
       geom_col(aes(x = diag_1, y = n, fill = readmitted)) +
       scale_fill_manual(values = ecols) +
       labs(x ="Primary diagnosis [category]", y = "Total counts")
p3 <- p3 + facet_wrap(~gender) +
       theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
p4 <- ggplot(agg) +
       geom_col(aes(x = diag_2, y = n, fill = readmitted)) +
       scale_fill_manual(values = ecols) +
       labs(x ="Secondary diagnosis [category]", y = "Total counts")
p4 <- p4 + facet_wrap(~gender) +
       theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

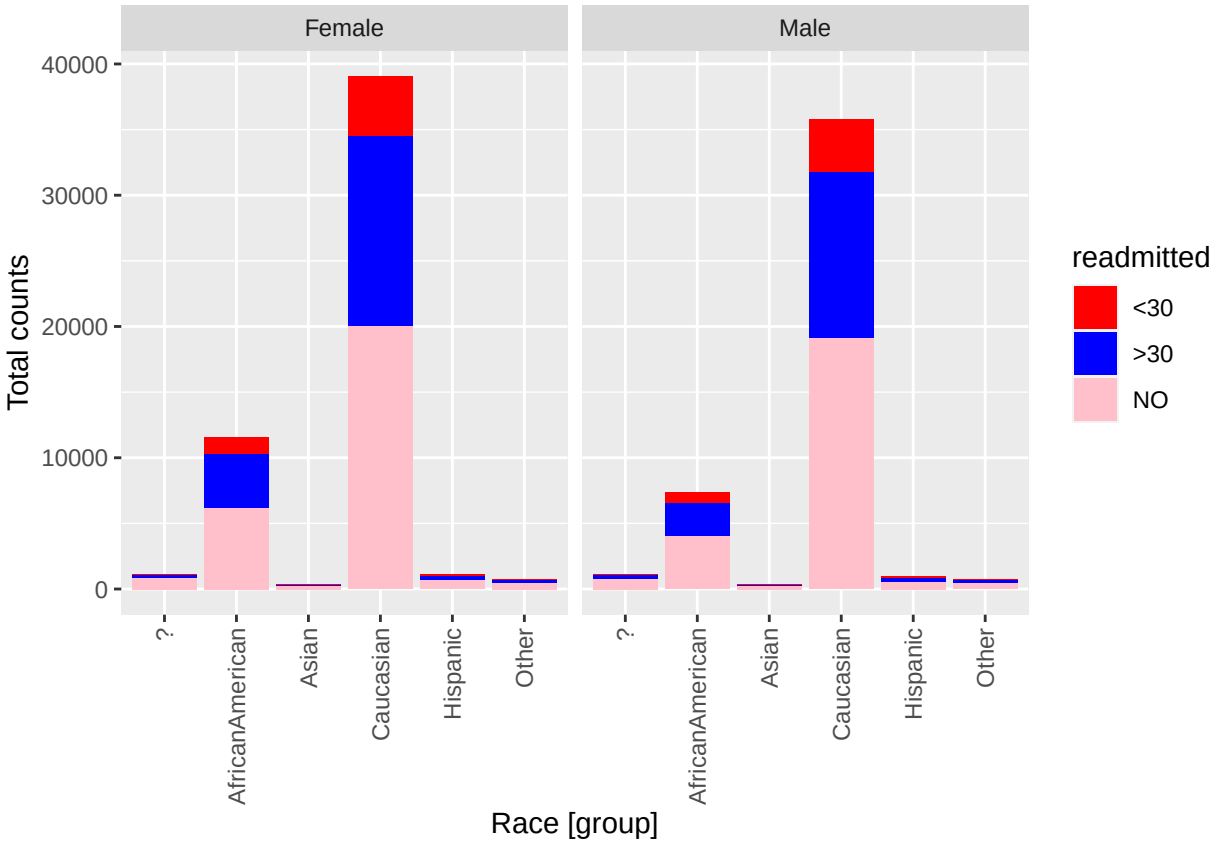


Figure 13: Readmitted distributed with race and divided into gender

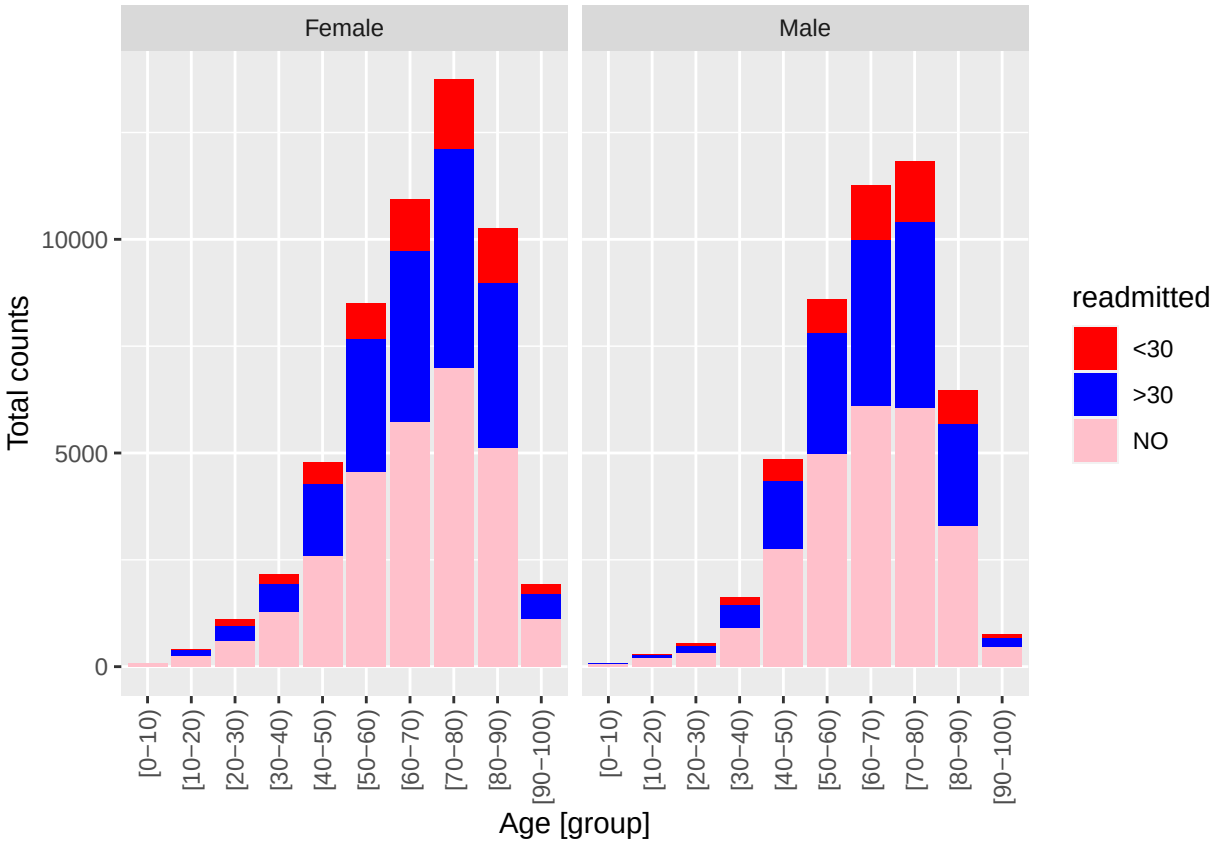


Figure 14: Readmitted distributed with age and divided into gender

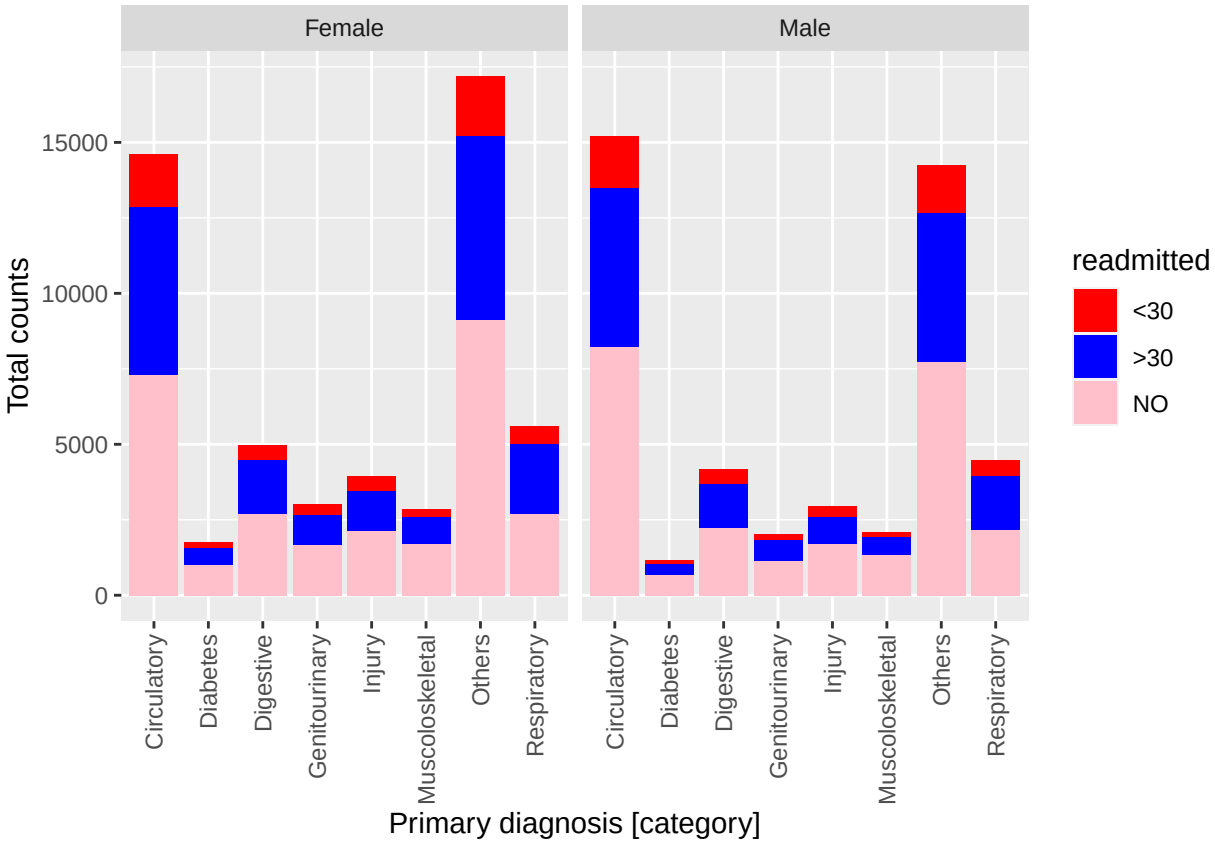


Figure 15: Readmitted distributed with primary diagnosis and divided into gender

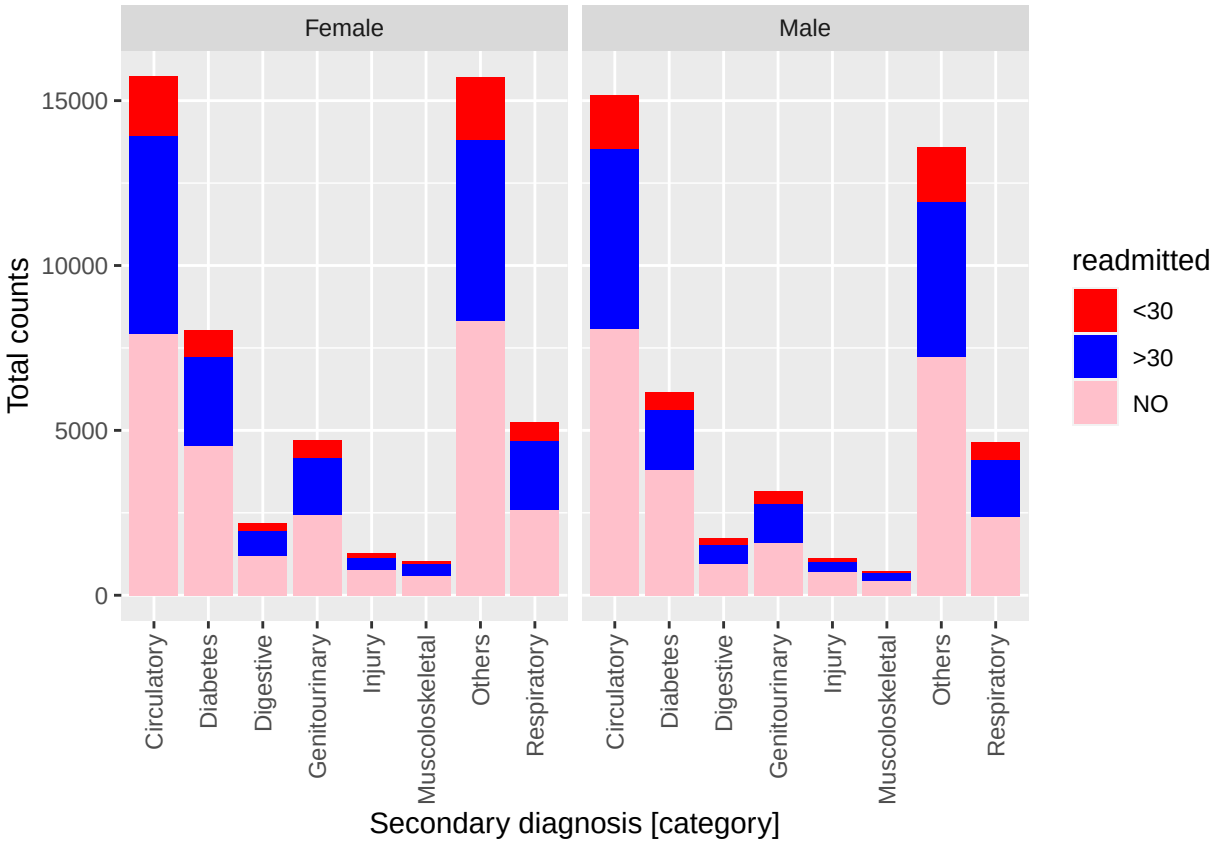


Figure 16: Readmitted distributed with secondary diagnosis and divided into gender

We observe the results of analyzing the readmitted attribute. We looked at readmitted with different demographics (gender, age, and race) and two of the three diagnosis attributes. Figure 13 shows that the Caucasian race has the most readmission rates in both genders. Observations made in figure 14 depict that as age increases, readmission rates also increase for females and males. This makes sense as immunity decreases with age, and the chance of recovery diminishes. Readmission rates decrease after the age of 80. The reasoning behind this might be death or transferring to a hospice or other facility.

Comparing the results in figures 15 and 16, we see a familiar trend: the increase of diabetes diagnoses between the primary and secondary diagnoses. This typically means that patients admitted were diagnosed differently, and over time, are getting a new diabetes diagnosis much quicker.

Since we have multiple diagnosis categories but are somewhat interested in diabetes only, we can alter these attributes to have only two valuations (‘diabetes’ and ‘other’). However, doing this can affect later analysis as we remove potentially valuable information.

2.3 Distribution - numeric attributes

We discussed many categorical data attributes, now turn to our few numeric data. It is important to have a good distribution between numeric data. If the range of distribution is too large, then normalization is necessary. We construct multiple histograms, which we get through the histogram.plotter() function.

```
# Define a histogram plotter to construct multiple figures really quick
histogram.plotter <- function(attribute_1, number, attribute_name){
  histogram <- ggplot() +
                geom_bar(mapping = aes(x=attribute_1)) +
                ggtitle(number) +
                xlab(attribute_name)
}

# Make a smaller dataset so that only the numeric data is collected
numeric.data <- select_if(encounter.data, is.numeric)
# Remove encounter and patient ID
numeric.data <- numeric.data[, -c(1, 2)]
smaller.codebook <- codebook[c(10, 13:18, 22), 2]
numeric.data.log <- log2(numeric.data + 0.1)

wrapper <- function(data){
  p <- list()
  for ( i in c(1:length(data)) ){
    p[[i]] <- histogram.plotter(data[, i], LETTERS[i], smaller.codebook[i])
  }
  return(p)
}
numeric.normal <- wrapper(numeric.data)
numeric.log <- wrapper(numeric.data.log)
```

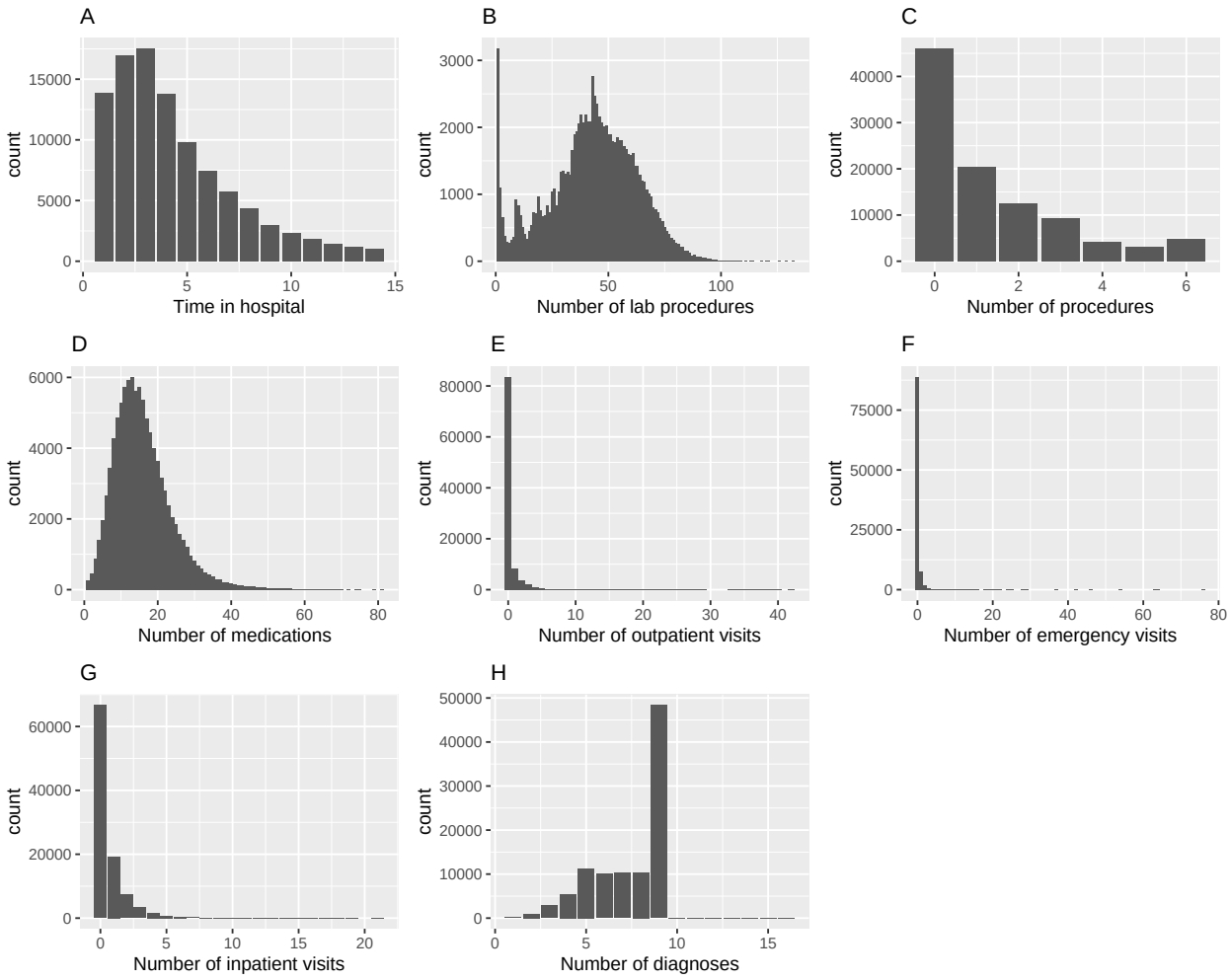


Figure 17: Numeric data presented in histograms without any normalization

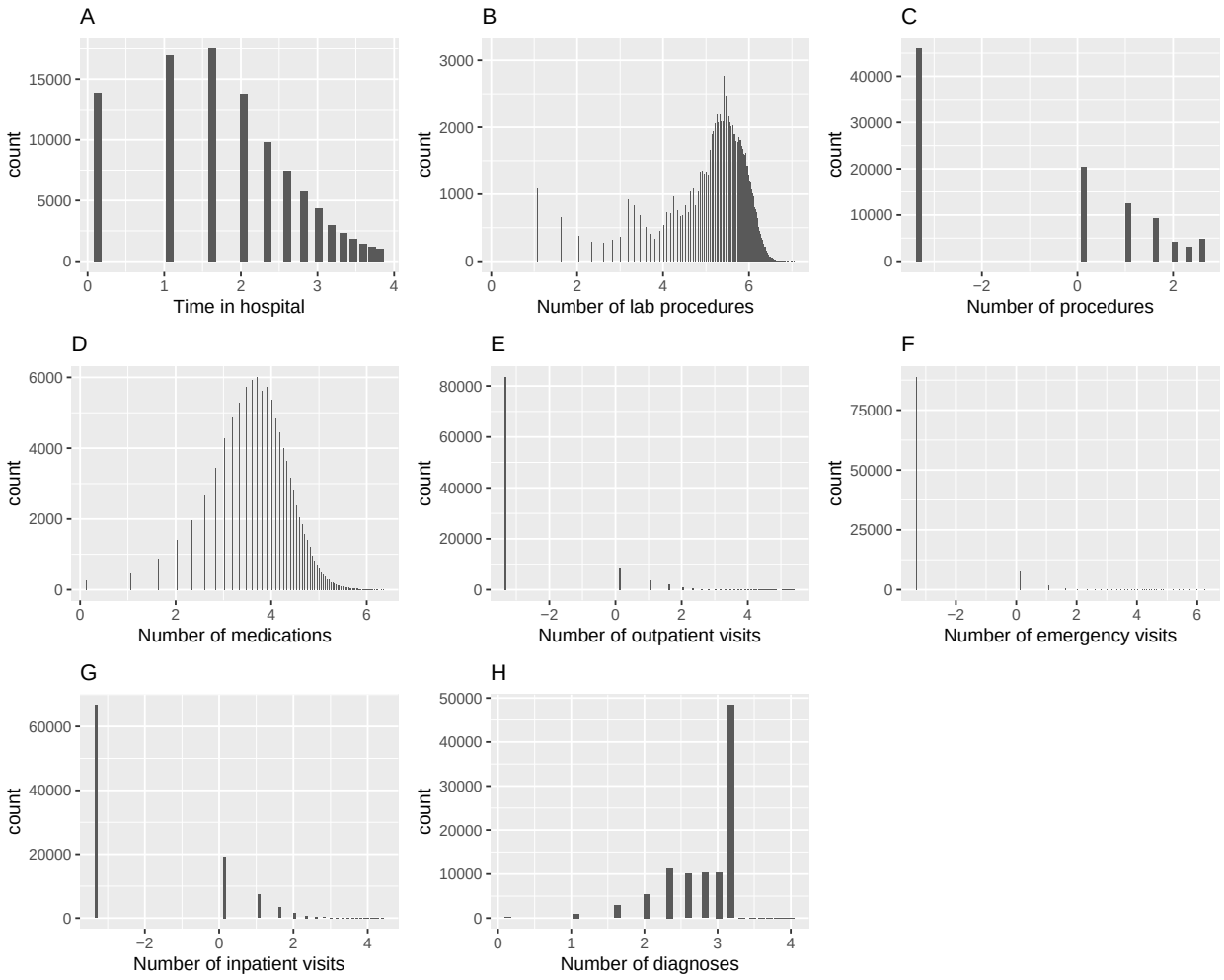


Figure 18: Numeric data presented in histograms on a log-2 scale

The results, depicted in figures 17 and 18, show that not all of the attributes are evenly distributed. Normalization might be necessary to correct the distributions of namely B, which makes a couple of weird spikes, E, F, and G, which are all -but B- number of visits of an encounter. We might want to introduce a new attribute, total_visits, which accounts for all visits made by one encounter. The method of normalization is still up for debate.

2.4 Correlation - numeric attributes

To discover any correlation between the numeric data in our dataset, we constructed a heatmap, which is depicted below. We discuss a possible correlation between age and time spent in the hospital and the potential relationship between the num_lab_procedures and num_medications.

```
library(reshape2)
melted.data <- numeric.data %>%
  cor() %>%
  melt()
```

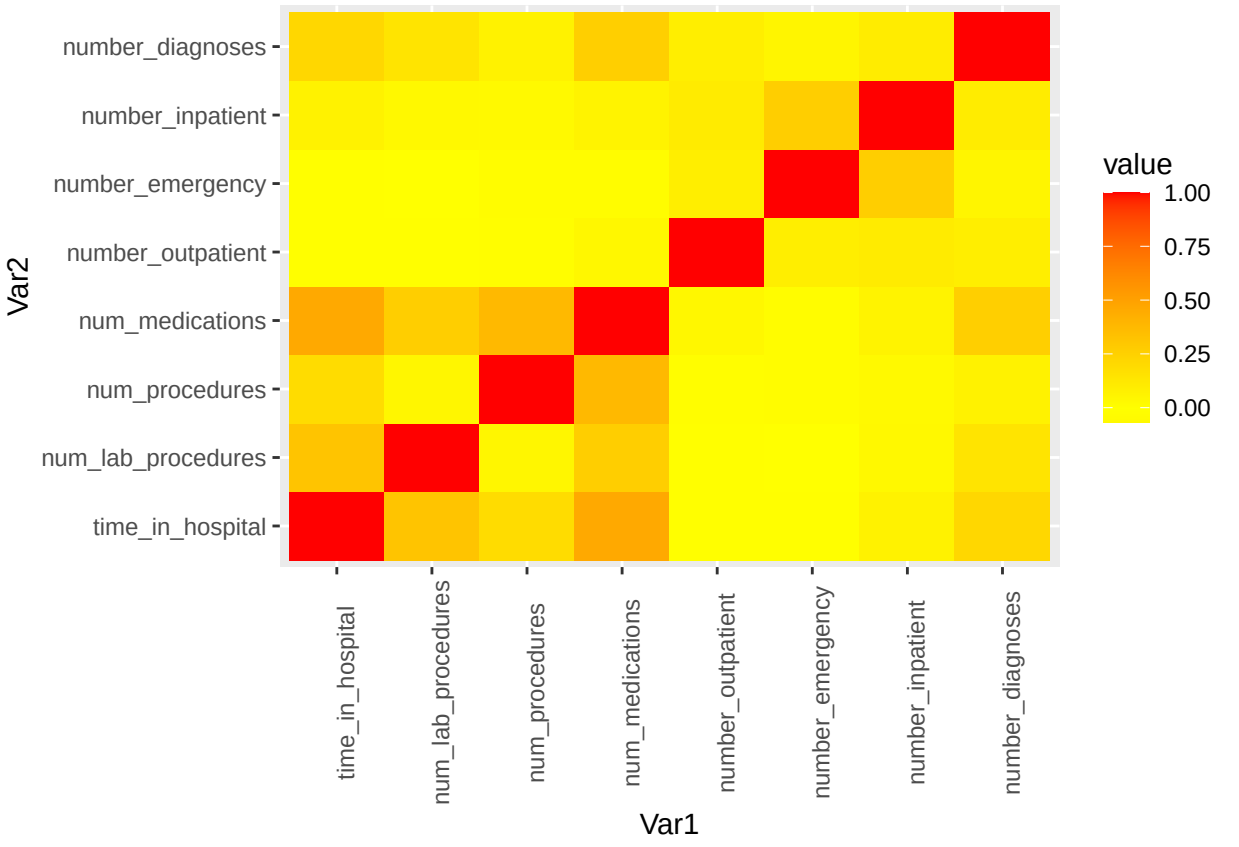


Figure 19: Heatmap of numeric attributes, with red depicted as a strong correlation, orange as a (small) correlation, and yellow as no correlation

Figure 20: Attribute time spent in hospital plotted against age group

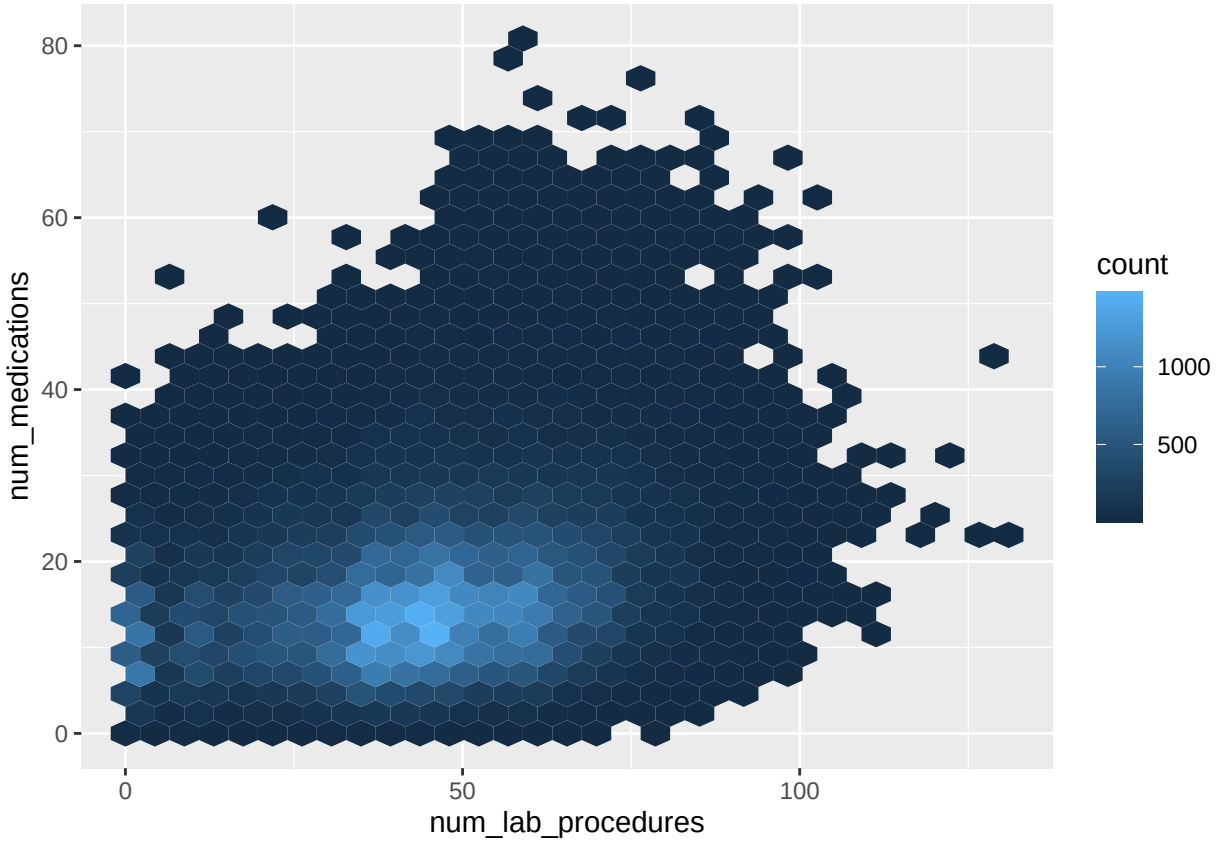


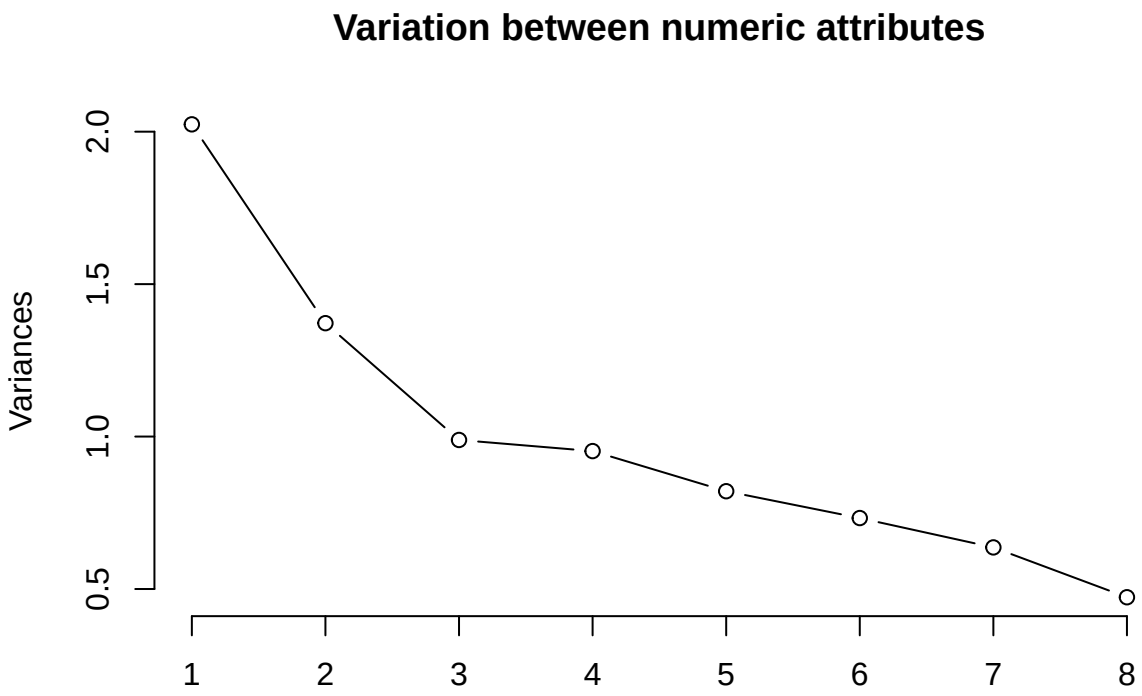
Figure 21: Attribute num_medications and num_lab_procedures plotted against each other

Our heatmap (shown in figure 19) shows a strong correlation as red, a small correlation as orange, and no correlation as yellow. The attributes that stand out are, for example, num_medications-time_in_hospital, num_procedures-num_medications, and hba1c_res-num_medications. As we can see, it is mostly the num_medication attribute that has some correlation. Others do not stand out that much. We can conclude that most of the attributes are non-correlated, at least for the numeric attributes.

Looking at figure 20, we observe that the age groups above [40-50) increases in total counts; consequently, the total time spent in the hospital before release also increases. We can, therefore, conclude that the time_in_hospital attribute is dependent on a patient's age. Finally, we zoom in on two attributes with a (small) correlation depicted in figure 19. Figure 21 shows the result of zooming in on num_medications and num_lab_procedures; the result shows that most encounters have a valuation between 15-20 medications and 40-60 amount of lab procedures. The result also shows a lot of outliers, which normalization could solve.

Next up, we look if our valuations of the class variable are clustered separately or if we have overlap between instances. This is important because it tells us something about how well we can predict based with the class variable. We investigate the variance between all numeric data compared to the class variable in figure 21. Here we see that the variation steadily declines per PCA run. If we look at the actual PCA, depicted in figure 22, we can see that there is a lot of overlap between attributes regarding the class variable. It means that we can expect some difficulty predicting with machine learning algorithms.

```
cc.pca <- prcomp(numeric.data,
                 center = TRUE,
                 scale. = TRUE)
```


Variation between numeric attributes
Variances
0.5
1.0
1.5
2.0
1
2
3
4
5
6
7
8

```r
library(ggbiplot)
ggbiplot(cc.pca, obs.scale = 1, var.scale = 1,
         groups = as.factor(encounter.data$time_in_hospital),
         ellipse = TRUE, circle = TRUE, alpha = 0.5) +
  scale_color_discrete(name = '') +
  theme(legend.direction = 'horizontal', legend.position = 'top') +
  xlim(-5, 20) + ylim(-10, 50)
```

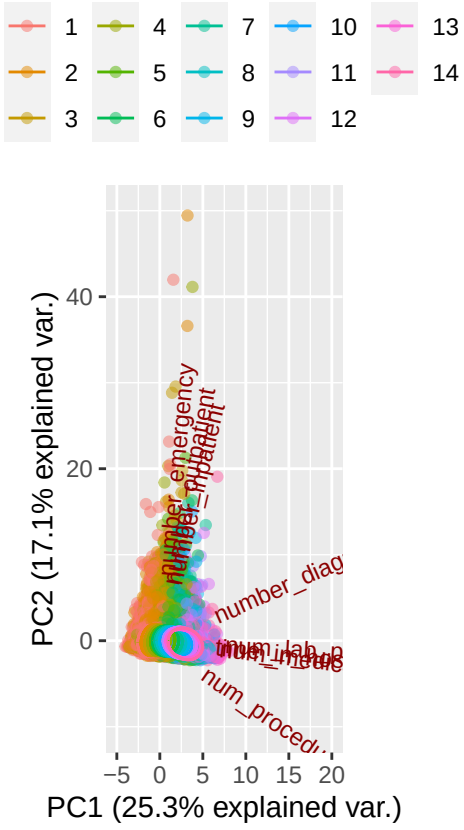


Figure 22: A PCA with all numeric attributes, plotted against the class variable time_spent_hospital, with fourteen different valuations

Since we adjusted the labels of the class variable, we need to perform a PCA once more. Figure 22 depicted a class variable with 14 different valuations; now we only have two. The results are depicted in figure 23. While we look at figure 23, we can see that the valuations still overlap quite a bit. But there is a clearer difference between them; we can see that a blue and red group are still a little separated from each other but they still overlap in the middle. This is most likely because of splitting the class variable into a group with lower and higher valuations.

```r
encounter.label <- encounter.data %>%
  mutate(time_in_hospital =
           case_when(time_in_hospital %in%
                       c('1', '2', '3', '4', '5') ~ 'brief',
                       TRUE ~ 'long'))

ggbiplot(cc.pca, obs.scale = 1, var.scale = 1,
         groups = as.factor(encounter.label$time_in_hospital),
         ellipse = TRUE, circle = TRUE, alpha = 0.05) +
  theme(legend.direction = 'horizontal', legend.position = 'top') +
  xlim(-5, 20) + ylim(-10, 50)
```

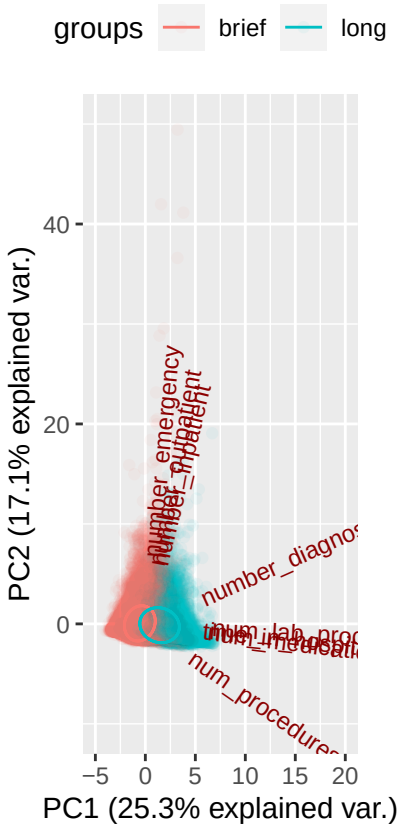


Figure 23: A PCA with all numeric attributes, plotted against the class variable time_spent_hospital, with two different valuations

3 A Clean Dataset

In this section, we will remove all unnecessary attributes and instances, as discussed in the EDA. Many filters and mutations have already been performed throughout the EDA, but not all have made it and seems to be scattered around - to give a more clear picture of the final, clean dataset, we perform all filtering, mutations, and relabeling that made it to the final dataset, again.

```
# Load in the data again to perform only filtering that is needed,
# discussed in the EDA
final.data <- read.table(
  file = 'datasets/data_diabetes_retrieved/diabetic_data.csv',
  sep = ',', header = TRUE)

final.data <- final.data %>%
  # Filter out every duplicate, based on patient number
  distinct(patient_nbr, .keep_all = TRUE) %>%
  # Relabel '?' to 'Missing'
  mutate(race=recode(race,'?'='Missing')) %>%
  # Remove instances with a missing labels and patients
  # who died in hospital
  filter(gender != 'Unknown/Invalid' & discharge_disposition_id != 11) %>%
  # Introduce the 'hba1c_res attribute, which is our class attribute
  mutate(hba1c_res=ifelse((A1Cresult=='None'), 'one', NA)) %>%
  mutate(hba1c_res=ifelse((A1Cresult %in% c('Norm', '>7'))
                          & is.na(hba1c_res), 'two', hba1c_res)) %>%
  mutate(hba1c_res=ifelse((A1Cresult=='>8') & (change=='No')
                          & is.na(hba1c_res), 'three', hba1c_res)) %>%
  mutate(hba1c_res=ifelse((A1Cresult=='>8') & (change=='Ch')
                          & is.na(hba1c_res), 'four', hba1c_res)) %>%
  # Introduce the new attribute icu_or_non
  mutate(admission_type_id =
           ifelse(admission_type_id %in% c(1, 2, 7),
                  'ICU patient', 'Non-ICU patient')) %>%
  mutate(admission_source_id =
           ifelse(admission_source_id %in% c(4, 7, 10, 12, 26),
                  'ICU patient', 'Non-ICU patient')) %>%
  mutate(discharge_disposition_id =
           ifelse(discharge_disposition_id %in% c(13, 14, 19, 20, 21),
                  'ICU patient', 'Non-ICU patient')) %>%
  mutate(icu_or_non =
           ifelse(admission_source_id == discharge_disposition_id &
                  admission_type_id   == discharge_disposition_id,
```

```
                  admission_source_id,
                  ifelse(admission_source_id == discharge_disposition_id,
                         admission_source_id, admission_source_id))) %>%
  # Change the label system in attribute diag_1
  mutate(diag_1 = case_when(diag_1 %in% c(390:459) ~ 'Circulatory',
                            diag_1 %in% c(460:519) ~ 'Respiratory',
                            diag_1 %in% c(520:579) ~ 'Digestive',
                            diag_1 %in% c(240:279) ~ 'Diabetes',
                            diag_1 %in% c(800:999) ~ 'Injury',
                            diag_1 %in% c(710:739) ~ 'Muscoloskeletal',
                            diag_1 %in% c(580:629) ~ 'Genitourinary',
                            TRUE ~ 'Others')) %>%
  # Set all character attributes to factor
  mutate_at(vars("time_in_hospital",
                 "hba1c_res",
                 "icu_or_non"), funs(factor)) %>%
  # Introduce a log2 scale on all numeric attributes
  mutate_at(.vars = names(select_if(., is.numeric)), ~ log2(. + 0.1)) %>%
  # Remove all redundant attributes
  select(-c('encounter_id', 'patient_nbr', 'weight', 'payer_code',
            'medical_specialty', 'diag_2', 'diag_3','admission_source_id',
            'discharge_disposition_id', 'admission_type_id',
            removal.names)) %>%
  relocate(time_in_hospital, .after = last_col())

# Write final dataset to datafile
write.csv(final.data, "datasets/base_datasets/normal_dataset.csv",
          row.names = FALSE)
```

4 Determine quality metrics relevancy

For the next part of our research, we will work on predicting the time spent in the hospital. Weka, a Java-based data analysis and algorithm interface, is used in this research. Its powerful interface makes it easy to work with comprehensive data modeling techniques.

Weka comes with quite a few ways to evaluate results gathered from a classifier. When predicting a class variable, Weka reports the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). In our case, this would be a correctly predicted duration of a hospital stay, a not correctly predicting the duration, a prediction that is classified as a correct duration is actually incorrect. An incorrect prediction is actually a correct one, respectively. These are essential values for evaluating predicted results after running a classifier.

Knowing these four crucial numbers, we can calculate quality metrics to determine how well a classifier works on the supplied dataset. The number of quality metrics to choose from is substantial, but we only need a couple to establish whether a classifier is worth-while. We look at the accuracy, precision, and recall of the results, the rates of true and false positives, F1-score measurement, determine a ROC area and construct a confusion matrix.

A confusion (or error) matrix represents the performance of an algorithm. It is a table with rows for the predicted instances, and the columns represent the instances of the actual class. The matrix reports on the number of FP, FN, TP, and TN. This representation allows for a more detailed analysis than a metric that specifies a certain value like accuracy. However, it can complicate when too many class variables need to be depicted. For example, our class attribute has 14 valuations and creates a confusion matrix of 196 (14x14) elements. It contains much information; therefore, we introduce other quantity metrics that work more specific and are less complex.

The number of instances correctly identified as either true positive or true negative is called the accuracy. This metric gives a peak in how "true" our classification is, or in other words, how many predictions are right. In our case, predicting the right amount of days that a patient spends in the hospital. It is essential to know how accurate our classification is as we want the result (prediction) to be as close to the 'true' value.

Precision and recall are also two important metrics. Precision is the balance between TPs and all other classified positive results (TP + FP). This metric's importance comes from the fact that we can check on the balance between truly correct predicted instances and falsely correct predicted instances. I.e., what proportion of the positive identifications was actually correct? It highlights how well our classifier performs on the data we fed to it. We want to determine the duration of hospital stays so that professionals can look at the differences between outcomes and potentially make an effective plan to reduce the duration of visits and thereby costs. Therefore, it is important to have a model with as little false, correct predictions, as this can affect the liability of future procedures to reduce the duration of visits. It can have less effect than was calculated, potentially increasing costs than was planned. Therefore, our goal is only to bring advice to hospitals and institutions if we can

confidently predict the duration of inpatient visits.

The recall is a metric that represents the sensitivity of an outcome. It looks at what proportion of actual positives was identified correctly (TP / TP + FN). Remember that False Negatives are actual correct predictions thought to be incorrect. So it reflexes on the number of missing correct predicted instances. This can also have an impact on the effectiveness of a procedure based on our model. It is the opposite of the message from precision. If False Negatives are with many - and the False Positives rate is low - then a procedure instituted on our model could have done more to reduce the duration of visits, but our model could not learn more about the true data and became more biased towards false predictions. It determines if a said procedure based on our model can reach full potential regarding its goals: reducing duration as much as possible to make health care more affordable.

An F1-score is the mean of the precision and recall; it measures the recall and precision metrics' effectiveness by calculating the mean. It can range from 0 to 1 (perfect precision and recall). In other words, the F1-score conveys the balance between precision and recall.

The last metric we introduce is the ROC area, which can be depicted in a ROC curve. The ROC curve is created by plotting the true-positive rate as a function of the false-positive rate. ROC analysis arranges a way to select optimal models and to discharge suboptimal ones. It helps with visualizing a complex confusion matrix. As we have 14 valuations in our class variable, it becomes a really complex job to interpret every bit of information depicted. A ROC area makes this part easier. The ROC area (AUC) is a value that represents how much a model can separate between the valuation of the class variable. Higher the AUC, the better capability of the model to distinguishing between the duration of visits.

5 Machine learning algorithm performances

We will perform a multitude of algorithms. We are interested in performing with Naïve Bayes, Simple Logistics, Nearest Neighbour (IBk), a decision tree in the form of a J48/C4.5, and a Random Tree. Besides, we also include ZeroR and OneR to measure baseline performance. Weka gives outcome for every class valuation (so in our case 1 to 14); we take the average sum of these outcomes, as it is better for comparison with other classifiers. The results are depicted in table 4.

```
# Results will be loaded in like this,
# but future code block will not be included due to repetitiveness
normal.result <- read.csv("datasets/dataset_results_weka/base/normal.csv",
                          sep = ";", fileEncoding="UTF-8-BOM")
kbl(normal.result, booktabs = T,
    caption = "Results from chosen classifiers on their base settings") %>%
  kable_styling(full_width = F, latex_options = "hold_position") %>%
  column_spec(1, bold = T)
```

Table 4: Results from chosen classifiers on their base settings

Algorithm	Accuracy	TP.Rate	FP.Rate	Precision	Recall	F.Measure	ROC.Area
ZeroR	17.82%	0.18	0.18	0.00	0.18	0.00	0.50
OneR	20.79% (v)	0.21	0.15	0.17	0.21	0.17	0.53
J48/C4.5	19.55% (v)	0.20	0.12	0.18	0.20	0.18	0.55
Naïve Bayes	20.76% (v)	0.21	0.12	0.19	0.21	0.20	0.65
IBk	16.16% (*)	0.16	0.12	0.16	0.16	0.16	0.52
Simple Logistics	23.60% (v)	0.24	0.14	0.20	0.24	0.20	0.68
Random Tree	17.20% (*)	0.17	0.12	0.17	0.17	0.17	0.53

We now observe the results shown in the table 4. The results of all involved classifiers show a not-so-great picture; the accuracy is very low across all classifiers, ranging from a mere 16,20% for Nearest Neighbor (IBk) to a high-low result of 23,55% with Simple Logistics. This situation is not ideal as - on average - only one-fifth of the class variable is correctly predicted. This is because the class variable is positively skewed, which means low-valuations are more present than higher values. In our case, this means patients spent just a couple of days in hospital rather than, for example, more than a week.

Next up is determining which classifiers are most worthwhile regarding our research goals. As stated earlier, accuracy and precision are the most important metrics. The best option seems to be Simple Logistics with an accuracy of 23.60% and a precision of 0,523 - both are best-performing in their category. Deciding on a runner-up appears more challenging as J48 and Naïve Bayes both perform well. Naïve Bayes scores better in every metric we present, but J48/C4.5 has better optimization options, while Naïve Bayes' result can be considered its final product. Besides, the accuracy difference is a mere 1.21%, not a significant difference. For this reason, we include both J.48/C4.5 in the coming sections.

5.1 Optimizing our classifiers

To determine the best possible algorithm, we need to play with some of our chosen algorithms' parameter settings. To accomplish this, we want to use Weka's Experimenter mode. This mode allows us to explore multiple parameter settings and rules, which are the most optimal. In other words, we want to crick up the accuracy of all algorithms involved. We will do this by exploring different meta-learners. The meta-learner CvParameterSelection was performed on J48/C4.5 and Simple Logistics to find the optimal parameter sets. Naïve Bayes was optimized manually, as it is a relatively simple algorithm with just a few settings. We present the results in the sections below. We perform all the classifiers on a cost-sensitive matrix in the earlier section and split the test set into ten pieces with cross-validation.

5.1.1 J.48/C4.5

First, we look at the parameter settings of the tree algorithm J48/C4.5. Here we have to deal with two parameters: the confidence factor parameter (C) and the minimal number of objects (M). The C value determines the amount of pruning performed; a small value corresponds to heavy pruning, whereas a large one to little pruning. The default is set to 0.25, and we will let the algorithm test with values ranging from 0.1 up to 0.5 in steps of 10. We set the limit to 0.5 as above; it will cause little to no pruning. [3] Parameter M's default is set to 2, which means that the minimum instances per leaf guarantees that at each split, at least 2 of the branches will have the minimum number of instances. Increasing these values means that leaves become bigger. We test with a range from 2 to 10 in steps of 1. The results are shown in table 5.

Table 5: Accuracy under different settings regarding J48/C4.5

Confidence.Factor	Accuracy.C	MinNumObject	Accuracy.M	Combination	Accuracy.C.M
0.10	20.95%	**2**	19.55%	**0.1/5**	22.28%
0.15	20.28%	**4**	20.56%	**0.1/10**	23.28%
0.25	19.55%	**5**	20.88%	**0.35/5**	20.57%
0.35	19.32%	**7**	21.34%	**0.35/10**	21.46%
0.45	19.11%	**10**	21.80%	**0.25/2**	19.55%

Looking at the table 5, we observe three columns of settings that caused a given accuracy, depicted beside every used setting, respectively. We first looked at the outcomes of the individual parameter setting. Thereafter we combined a few of these settings to find the most optimum parameter settings. The confidence factor shows a pattern where lower values give higher accuracy; it means that we perform more pruning (reducing the decision tree's size), which leaves us with better outcomes. Nevertheless, we have to be careful with this parameter to overfit our model on the training data we provide. Therefore, we want to consider this value carefully. As for parameter M, we see an opposite pattern happening; increasing this value comes with better accuracy. With the default setting (a value of 2) returning 19.55% accuracy, we see an increase of 1.33% percentage points as we set this to 5.

With a value of 10, the accuracy becomes its most optimal point yet: 21.80%. In combining both optimal settings, we get an accuracy of 23.28%, also the highest in the Combination column. Optimizing this algorithm's settings, we see an increase of 3.73% at the finish line. Truthfully, it is not a significant increase from its baseline performance, but it is still a better result. Considering we performed cross-validation on the confidence factor and the outcome looks promising not to be overfitted, we can take these settings and conclude that this is the most optimum form of J48 regarding our research.

5.1.2 Simple Logistics

Now we turn to Simple Logistics (SL). SL has the advantage of built-in attribute selection. This means that it stops adding SimpleLinearRegression models when the cross-validation classification error no longer decreases. [4] This algorithm has quite a few parameters to its disposal, but we will not refrain from checkout every one of them, as not each one is relevant. Here we deal with the following set of parameters: useAIC (A), heuristicStop (H), maxBoostingIterations (M), numBoostingIterations (I), and weightTrimBeta (W). A determines when to stop iterations of adding models. This is an alternative to cross-validation. H is a value that decides whether a run is stopped if no new error minimum has not been reached in the last iteration of H. Weka's advice is to use the default value, but we are interested in its effects when we change this value. M sets a maximum number of iterations for LogiBoost. The default is set to 500, and it is advised to use a higher number in bigger datasets, which is the case for us. I is a counterpart of M in which sets a fixed number of iterations for LogiBoost. If < 0, the number is cross-validated, or a stopping criterion on the training set is used, which is the default. Moreover, W states that only instances carrying (1 - W)% of the previous iteration weight are made available for the next one, resulting in fewer instances but maybe a more accurate model. We depict our findings in the table 6.

Table 6: Accuracy under different settings regarding Simple Logistics

Settings	Accuracy	TP.Rate	FP.Rate	Precision	Recall	F.Measure	ROC.Area
Default	23.60%	0.236	0.138	0.203	0.236	0.205	0.677
A	23.64%	0.236	0.137	0.205	0.236	0.207	0.678
H (10)	23.43%	0.234	0.140	0.199	0.234	0.200	0.676
H (100)	23.58%	0.236	0.138	0.203	0.236	0.205	0.678
I (100)	23.63%	0.236	0.137	0.205	0.236	0.207	0.678
I (250)	23.63%	0.236	0.136	0.207	0.236	0.208	0.678
M (400)	23.59%	0.236	0.138	0.203	0.236	0.205	0.677
M (600)	23.59%	0.236	0.138	0.203	0.236	0.205	0.677
M (1000)	23.59%	0.236	0.138	0.203	0.237	0.205	0.677
W (0.3)	20.44%	0.204	0.170	?	0.204	?	0.635

All parameters involved do not have a big impact on improving the accuracy of the classifier. Combining parameters did not help either, but their results are not shown. This proves that the SL algorithm already delivered the closest to the most optimal situation possible. As we look back to the table 6, we already saw that this algorithm had a relatively high accuracy with default settings compared to others. Also, we already stated that this algorithm has the ability to stop adding models when cross-validation errors no longer occur; in other words, it stops when it finds the most optimum situation. For these reasons, we do not adjust parameters and continue with the default settings.

5.1.3 Naïve Bayes

As discussed earlier, Naïve Bayes is a relatively simple algorithm with just a few parameter settings, and therefore we optimized it manually. We had two options to research and present our findings in table 7, depicted underneath. These options are: 1) using a kernel density estimator rather than a normal distribution (K), and 2) using supervised discretization (D). The K option is a way to estimate the probability density function of a variable; in a sense, it is smoothing out the data a bit, in a different way than via the normal distribution. Alternatively, it weighs observations differently depending on how far from a certain point we are evaluating. The D parameter considers the class value, whereas the default, unsupervised discretization, does not. Therefore, we expect better results as the filter will break our attributes into bins that provide the most information about the class variable. We cannot combine the two parameters as that would create conflict.

Table 7: Accuracy under different settings regarding Naïve Bayes

Settings	Accuracy	TP.Rate	FP.Rate	Precision	Recall	F.Measure	ROC.Area
Default	20.76%	0.208	0.118	0.192	0.208	0.197	0.647
K	23.25%	0.233	0.131	0.203	0.233	0.208	0.669
M	22.61%	0.226	0.127	0.199	0.226	0.205	0.668

We now observe the table 7, where both introduced parameters positively impacted the accuracy and all other metrics involved. Namely, the K parameter increases the accuracy a few percentage points, so does the M parameter but to a less extent. In other areas, the results are even closer between both parameters: 0.203 precision for K and a score of 0.199 concerning M. But looking at the bigger picture, we conclude that K is the better option to go along with. Still, now optimized, Naïve Bayes still scores less on every metric than SL performed in its baseline. We will discuss which of the three classifiers is the final one after some more optimization.

5.2 Selecting attributes

Weka contains a meta-learner called 'AttributeSelectedClassifier.' This meta learner allows us to evaluate if a given classifier performs better without certain attributes. It does this in two steps: 1) dimensionality reduction through attribute selection and; 2) passing on to a given classifier. [5] We have the option to rank attributes hinged on their performance regarding the accuracy or any other metric. We will look at three different evaluators of this classifier to research if we are better off without any of our remaining attributes. These three different ones are 1) CorrelationAttributeEval, 2) InfoGainAttributeEval, and 3) GainRatioAttributeEval. Before dropping any attribute, we investigate the influence of a low-performing attribute. We also take their performance over the board of evaluators we are going to use into account. This can vary between evaluators and is essential to investigate.

5.2.1 Correlation

We look at the CorrelationAttributeEval, which calculates the correlation between each attribute and the class variable. When we run an evaluator, we look for values that outline a moderate-to-high or negative correlation, ranging from 1 to -1. We want to avoid attributes with a near-zero or zero correlation, as these attributes can potentially harm our model's performance. Therefore, we want to drop these attributes. The Ranker search method will be used here.

We observe the results in the table 8, and it is not great, to say the least. Our highest scoring attribute is num_mediations with a score of 0.12651, and our least-scoring is rosiglitazone with a mere 0.000389. In most experiments, it is usual to use a cut-off score of 0.05 for an attribute's relevancy. If we would do this, we are left with four of the 23 discussed attributes. That would not be worth it if we want to continue with these research goals. If we proposed another cut-off of, for example, 0.01, we would lose eight attributes instead of 19. We like to look at the results of other evaluators before making a final decision.

Table 8: Correlation between all independent attributes

Attribute	Correlation
num_medications	0.12651
num_lab_procedures	0.07422
number_diagnoses	0.06756
num_procedures	0.05114
change	0.03278
insulin	0.02285
diabetesMed	0.02270
hba1c_res	0.02034
diag_1	0.02014
A1Cresult	0.01970
number_inpatient	0.01868
icu_or_non	0.01729
readmitted	0.01707
age	0.01459
gender	0.01257
metformin	0.00947
glyburide	0.00922
glipizide	0.00840
number_outpatient	0.00795
race	0.00472
number_emergency	0.00418
pioglitazone	0.00402
rosiglitazone	0.00389

5.2.2 Info Gain

Next up is an interesting evaluator called InfoGainAttributeEval, which looks at the information gain or entropy for each attribute for the output/class variable. Calculated values can range between 0 (no information) and 1 (maximum information). Attributes with a high valuation contribute more to the model than lower values do, as they reduce entropy more. There is the potential of removing attributes with a low valuation that could increase accuracy. Like we did in the correlation evaluator, we use the Ranker search method once more. We depict our results and findings in the table 9.

As we have seen in the section about correlation, we see that the same attributes perform worse than others. Some even perform so bad that they get a score of zero - number_emergency and number_outpatient - which shows that at least these two attributes are up for removal. Deciding on a cut-off value is difficult; what has considered a 'normal' cut-off value means, again, that we would throw away at least 80% of the attributes. In the case of 0.05, we would be left with only two attributes and with 0.01 eight attributes. We can look at the effects of removing sets of attributes at a time and rerun our classifiers to determine the effects.

Table 9: Info gain regarding each independent attributes

Attribute	Info.Gain
num_medications	0.18362
num_lab_procedures	0.10033
number_diagnoses	0.04923
num_procedures	0.03651
diag_1	0.02377
age	0.01880
insulin	0.01602
change	0.01094
hba1c_res	0.00639
diabetesMed	0.00507
A1Cresult	0.00500
readmitted	0.00452
glipizide	0.00380
number_inpatient	0.00350
glyburide	0.00328
icu_or_non	0.00317
metformin	0.00249
pioglitazone	0.00163
gender	0.00147
rosiglitazone	0.00135
race	0.00131
number_emergency	0.00000
number_outpatient	0.00000

5.2.3 Gain Ratio

This evaluator looks a lot like the one we just discussed. It is an enhancement of the InfoGain evaluator, with a normalized score. This method measures the significance of attributes concerning the class variable based on a gain ratio in more detail. A higher gain ratio means it reduces more entropy. We again use the Ranker search method. The results are shown in table 10.

Again, we see an occurring trend: the same five attributes - num_medications, num_lab_procedures, num_diagnosis, num_procedures, and change - perform the best with each evaluator. The same can be said about the other side. The attributes number_emergency and number_outpatient are both worst-performing once more. We will remove these attributes, at least. In the next section, we will discuss the attributes we will remove and which ones are here to stay.

Table 10: Gain ratio regarding each independent attributes

Attribute	Gain.Ratio
num_medications	0.06346
num_lab_procedures	0.03877
num_diagnoses	0.02652
num_procedures	0.01968
change	0.01102
insulin	0.00945
diag_1	0.00937
age	0.00707
number_inpatient	0.00671
hba1c_res	0.00671
diabetesMed	0.00640
glipizide	0.00605
glyburide	0.00566
A1Cresult	0.00520
pioglitazone	0.00400
rosiglitazone	0.00363
readmitted	0.00353
icu_or_non	0.00322
metformin	0.00292
gender	0.00147
race	0.01120
number_emergency	0.00000
number_outpatient	0.00000

5.2.4 Final decision

As already stated, we will remove the attributes number_emergency and number_outpatient. Considering if we needed to remove more attributes is a more difficult decision, almost all involved attributes do not score sufficient even to come close to the considered normal cut-off value of 0.05 or, in some cases, 0.01. Therefore, we want to explore another option that involves relevancy in the information an attribute gives us. For example, the attribute gender contains critical information regarding further research goals, and being able to separate cases on gender is vital in medical studies but has a low score in every attribute evaluation. Even though it has a low score, we want to retain it because of its content. Another example is metformin, which also scores low but is not of importance as diabetesMed. When applying this logic across the broad of attributes, we select candidates to remove. These are, in random order: metformin, diabetesMed, rosiglitazone, pioglitazone, glipizide, and glyburide. This leaves us with 16 attributes, including the class variable. We performed all our classifiers again on the new state of the dataset. We only consider accuracy. The results are depicted in table 11.

We now observe table 11, and we can see that removing said attributes improves only the accuracy and other metrics of Naïve Bayes and J48/C4.5; Simple Logistics performs slightly

Table 11: Results from removing certain attributes (new) versus not removing them (old)

Metric	Old.NB	New.NB	Old.SL	New.SL	Old.J48	New.J48
Accuracy	20.76%	23.42%	23.60%	23.47%	19.55%	23.42%
TP.Rate	0.21	0.234	0.24	0.235	0.20	0.234
FP.Rate	0.12	0.129	0.14	0.140	0.12	0.134
Precision	0.19	0.207	0.20	0.200	0.18	0.202

worse with fewer attributes. Taking the best-performing version of a classifier seems to the best option. So in the case of taking Simple Logistics as our final model, we need to decide whether we want to keep the attributes we removed that concluded in better accuracy for both other models or get rid of them here. The attributes that were removed contained very little important information relative to others. Never mind that they resulted in better accuracy in the case of Simple Logistics - but not with a big margin (just 0.13%). Therefore, we are going to continue with fewer attributes for each classifier.

5.3 Confusion matrices

In this section, we look at the confusion matrices of the best-fitting versions of our chosen classifiers. Since we have a class variable with 14 values, a matrix would have a 14x14 structure. For the reason of that complexity, we have always looked at the weighted average when comparing models. Now that we want to make a final decision on which model we are going to use, with the fact that all our models' performances are relatively close to one another, it can help to look at the details for making a final decision.

5.3.1 Naïve Bayes

We will look at the outcome regarding Naïve Bayes first. The confusion matrix is shown in the table 12. When analyzing a confusion matrix, it is essential to know that the (longest) diagonal line represents all true predictions. Off-diagonal elements are the ones that are mislabeled by a classifier. Naturally, the higher the diagonal values, the better as it indicates many correct predictions (i.e., a high TP rate). Naïve Bayes performed relatively well with an accuracy of 23.42%.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	<– classified as
5036	2811	1968	450	153	46	21	11	3	2	1	0	0	0	a = 1
3523	4023	3037	931	426	149	50	41	33	4	1	0	2	1	b = 2
2334	3239	4250	1468	730	263	95	111	49	9	4	0	2	1	c = 3
1292	2050	3159	1416	818	261	160	233	63	12	8	0	3	1	d = 4
723	1138	2054	1118	815	314	176	287	76	22	13	0	10	2	e = 5
444	719	1510	825	635	330	186	294	103	27	13	1	6	3	f = 6
273	476	1035	669	523	308	171	321	87	30	15	0	22	7	g = 7
208	260	694	427	392	235	148	298	98	59	28	2	16	7	h = 8
95	155	429	281	255	177	135	248	80	41	22	2	12	9	i = 9
60	109	324	207	184	145	109	224	73	41	30	1	10	6	j = 10
46	81	241	170	155	134	80	170	53	42	20	0	13	14	k = 11
39	54	182	123	104	94	64	139	58	34	21	2	6	6	l = 12
40	36	127	85	96	70	67	138	50	24	24	1	11	2	m = 13
24	31	122	66	69	61	46	106	49	33	26	2	12	4	n = 14

Table 12: Confusion matrix based on the optimal settings regarding Naïve Bayes

When we observe the results, we see - again - that our data is positively skewed as the total instances decrease while the time in hospital spent increases. This is exactly the problem when predicting the exact time a patient will spend in the hospital; we have much data regarding short visits and not so much with longer visits. Tackling this problem means more instances for longer visits, as we get more information regarding these instances. Something that we lack right now. Besides, the accuracy and precision per class variable also decrease with increasing total time spent in the hospital. For example, we only have four correct predictions when looking at 14 days spent in the hospital. Compare that to 5.036 instances for one day in the hospital (a).

5.3.2 Simple Logistics

Next up is the confusion matrix retrieved from the Simple Logistics classifier, depicted in table 13. Simple Logistics had a better accuracy without removing any initial attributes, but we decided to use the one with fewer attributes regardless. This was that supplying unnecessary attributes for a small increase in accuracy is not worth the burden. Therefore, we have a Simple Logistics classifier that performs with a 23.47% accuracy.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	<– classified as
4556	3091	2566	217	53	7	11	1	0	0	0	0	0	0	a = 1
3127	4167	4099	579	163	45	32	8	0	1	0	0	0	0	b = 2
2029	3345	5645	1027	325	93	70	20	1	0	0	0	0	0	c = 3
1150	2092	4375	1098	446	150	101	55	4	4	0	0	1	0	d = 4
646	1214	2920	1004	531	149	179	70	18	7	5	0	4	1	e = 5
432	735	2087	848	492	175	180	103	24	11	5	0	2	2	f = 6
256	518	1478	681	422	202	190	124	31	23	9	0	2	1	g = 7
203	316	955	479	341	157	233	115	35	30	5	0	2	1	h = 8
92	163	606	330	261	146	197	96	27	14	4	0	3	2	i = 9
60	125	454	233	218	111	181	74	36	16	8	0	3	4	j = 10
39	86	361	181	181	102	129	72	28	19	8	0	11	2	k = 11
34	57	255	142	133	78	115	58	24	16	5	0	5	4	l = 12
34	43	197	77	122	58	119	54	31	14	12	1	6	3	m = 13
18	35	167	79	85	67	91	43	29	21	6	0	9	1	n = 14

Table 13: Confusion matrix based on the optimal settings regarding Simple Logistics

The presented matrix looks quite familiar to Naïve Bayes - positively skewed and not many predictions regarding higher valuations. We can see but is a little difficult to do so, that the number of correct predictions regarding a is significantly less than seen in Naïve Bayes. The numbers stand at 4.556 for Simple Logistics and 5.036 regarding Naïve Bayes. In contrast, the true positives of other class variables are higher across the board... but this effect wears off when the class variables increase. This effect could be the result of the weight Simple Logistics puts on attributes. It is a more complex classifier than Naïve Bayes. However, it does not result in a better-performing classifier.

5.3.3 J48/C4.5

Last but not least is J48/C4.5, where accuracy of 23.42% was achieved after attribute selection. The confusion matrix of the classifier is shown in the table 14.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	<– classified as
5071	2593	2258	315	122	72	39	18	3	2	4	2	1	2	a = 1
3344	3746	3973	570	275	147	86	35	19	9	13	2	2	0	b = 2
2137	3095	5346	922	481	256	159	71	38	20	19	4	4	3	c = 3
1257	1906	4193	882	541	279	204	103	42	26	19	10	8	6	d = 4
643	1181	2787	697	517	367	244	136	60	52	33	9	11	11	e = 5
440	730	1968	573	444	352	269	134	64	42	36	16	16	12	f = 6
284	508	1417	438	395	282	230	169	79	58	43	12	14	8	g = 7
186	358	884	301	304	228	216	181	56	63	55	12	22	6	h = 8
95	191	550	199	205	190	182	132	59	47	45	10	22	14	i = 9
71	138	402	174	169	129	130	126	38	40	48	25	21	12	j = 10
55	92	294	120	141	114	116	107	58	44	35	13	17	13	k = 11
36	73	222	99	107	93	84	80	35	31	33	11	13	9	l = 12
37	49	178	74	73	64	84	78	33	39	29	11	13	9	m = 13
27	46	143	59	59	77	44	69	26	35	36	6	13	11	n = 14

Table 14: Confusion matrix based on the optimal settings regarding J48/C4.5

Again, we see almost the same results in the performance. There are some differences between individual class variables' performances, but not many dissimilarities than the other discussed classifiers. Another point regarding J48 is that the tree produced is massive and too complex to be informative. This is also related to the number of attributes and class variables. The reason given can be legitimate to renounce this classifier because of the great similarities between all involved classifiers.

Observing the chosen classifiers' confusion matrices did not result in a breakthrough in choosing the final model. As we already established before discussing the matrices, is that the classifiers perform almost perform the same. There are some differences in the accuracy per class variable, but this is not enough overall. Having an accuracy of around 23% is just not acceptable for a model that needs to exactly predict the time spent in the hospital. This is mostly due to the skewness of our data. For example, having more instances regarding longer visits would help determine the difference between the total time spent in the hospital. A model predicting the exact length in days is too good to be true; therefore, we want to explore different scenarios where a more accurate model is helpful. Furthermore, of course, thinking about methods to increase accuracy. This is exactly what we are going to do in the next sections.

6 Change of plans

As we have seen in the presented results in the sections above, our model's accuracy is softly said, not great. We top at an accuracy of 23.60%, which is generally seen as too low to be used in daily life situations. In this section, we explore more abrupt ways to increase the accuracy; think of restructuring or removing instances from our class variable, which we did not want to do first. Also, we could change the research question to be more applicable to the new situation.

6.1 Effects of removing instances

```
count.data <- final.data %>%
  group_by(time_in_hospital) %>%
  tally()

instance.data <- final.data %>%
  filter(!time_in_hospital %in% c(11, 12, 13, 14))

write.csv(instance.data,
          "datasets/base_datasets/less_instances.csv",
          row.names = FALSE)
```

As we have seen, there is no magic trick that would increase accuracy without severely adjusting our data. Think about removing instances with higher valuation, reconstructing instances, or manually giving weight to high values to create a more normal-looking distribution. We already performed a cost-sensitive classifier to give more weight to values with fewer instances. Awarding even more weight could create a false image of what is really going on. Removing instances with higher values that are less covered is the last thing we want to do. Then we need to draw a line somewhere: what values can stay and which ones need to go? To illustrate the effects of removing high-value instances, we consider removing instances with a valuation ranging from 11 to 14; therefore, we would remove 3567 instances, which translates to a loss of around 6%. After removal, we are left with 66.871 instances.

Now that we have a new, temporary dataset, we turn to Weka once more and see if the rate of correctly classified instances has improved. The short-hand answer to that question is yes, it has improved, but not substantially enough to consider removing these instances for our final model. We performed the J48 and Simple Logistics classifiers again and observed an improvement of 1,13% compared to our first run (19,81% versus 20,94%) for J48. We noticed a similar trend for Simple Logistics. Here, the improvement is just 1,33%. The results are depicted in table 15. Removing 6% of our instances for such little improvement in accuracy overall is not worthwhile; therefore, we can safely conclude that this method is not what we are looking for in determining a better accuracy.

Table 15: Results from chosen classifiers on their base settings and removing extra instances to improve metrics

Algorithm	Accuracy	TP.Rate	FP.Rate	Precision	Recall	F.Measure	ROC.Area
ZeroR	18.77%	0.19	0.19	0.00	0.19	0.00	0.50
OneR	21.94% v	0.22	0.16	0.20	0.22	0.18	0.53
J48/C4.5	21.11 (v)	0.21	0.13	0.20	0.21	0.20	0.55
Naïve Bayes	21.94% (v)	0.22	0.13	0.21	0.22	0.21	0.64
IBk	17.23% (*)	0.17	0.13	0.17	0.17	0.17	0.52
Simple Logistics	24.73% (v)	0.25	0.15	0.22	0.25	0.22	0.67
Random Tree	18.23%	0.18	0.13	0.18	0.18	0.18	0.53

6.2 A new research goal

Another option to improve accuracy could be a relabeling of the attribute. This method has the upside of retaining all instances. This method's downside is that we create bigger data groups, thereby removing the chance of exactly determining how many days were spent in the hospital. While we cannot determine exactly how many days a patient spent in the hospital, we can separate visits based on classification, such as "a brief visit" or "a longer visit". So, classification is still possible but is less specific. We considered a relabeling with two different valuations:

1. Time spent in hospital is within or equal to 5 days (brief visit); and
2. Time spent in hospital is longer than five days (long visit).

In the next code section, we mutate the time_in_hospital attribute, as described above. We also show the implications of relabeling in figure 22 . As we can see, we now have two classes comprising of 55.000 and 19.000 instances, respectively.

```
final.data.label <- final.data %>%
  mutate(time_in_hospital =
           case_when(time_in_hospital %in%
                       c('1', '2', '3', '4', '5') ~ 'brief',
                       TRUE ~ 'long')) %>%
# Reshape data so that class variable is the last column
# We did this manually using the Explorer, but now we use the Experimenter,
# and the only way to appoint a class variable is having it be the last column.
  relocate(time_in_hospital, .after = last_col())

write.csv(final.data.label, "datasets/base_datasets/relabeled.csv",
          row.names = FALSE)

# Add legends and introduce a two-in-one plot
```

```
p1 <- ggplot() +
  geom_bar(mapping = aes(x=final.data$time_in_hospital)) +
  ggtitle("Time in hospital (1-14 days)") +
  xlab("Days")

p2 <- ggplot() +
  geom_bar(mapping = aes(x=final.data.label$time_in_hospital)) +
  ggtitle("Time in hospital (two values)") +
  xlab("Classification")
```

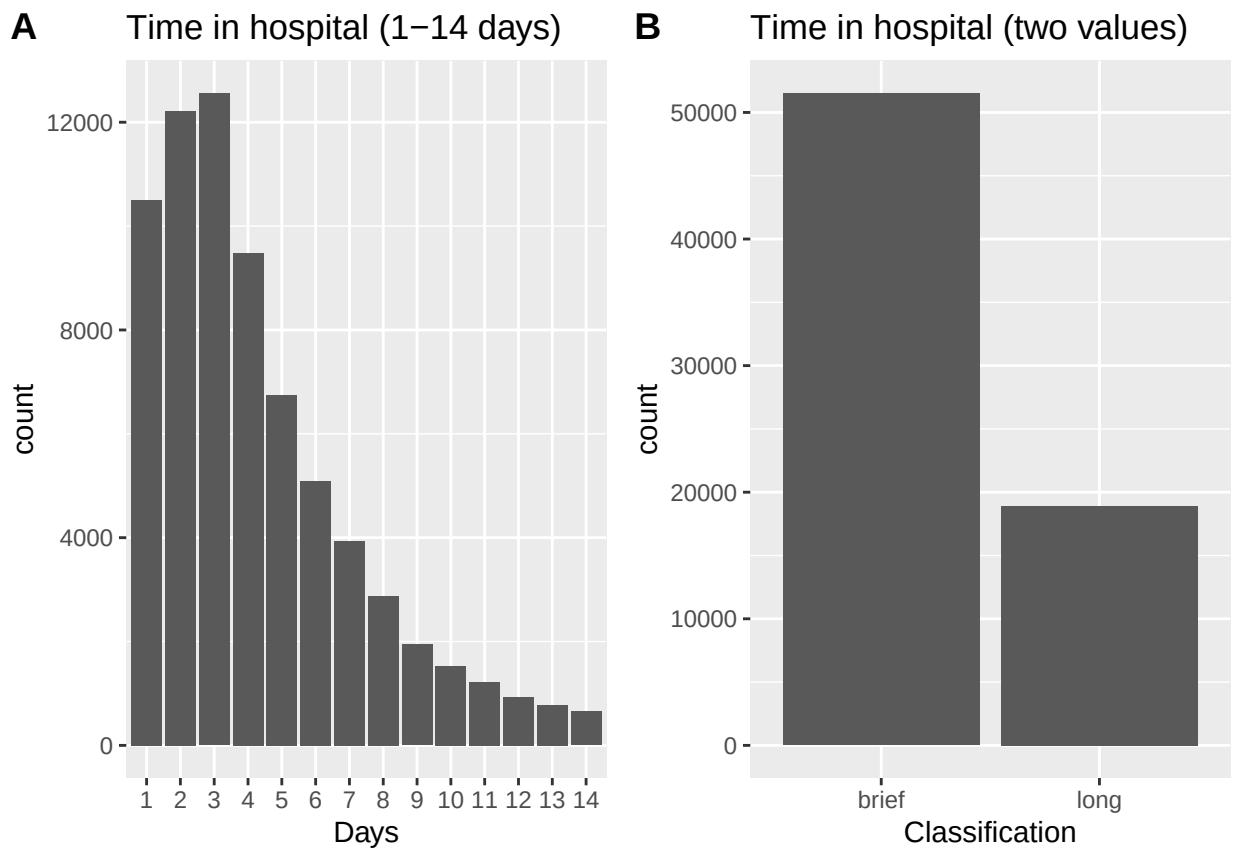


Figure 24: Difference between class variables labeling

Now that we have a new dataset, we turn to Weka once more to analyze the new dataset. The results of this are shown in table 16.

Having obtained results regarding both datasets, we can determine the differences. As expected, accuracy rose across the broad of classifiers; most experienced an increase of at least 50 percentage points, except for IBk, which rose 40 percentage points. IBk already had the least sufficient outcome in previous Weka runs, so logically we would not have expected it to become best-performing. When considering taking a final dataset to the next stage in our research, we must weigh if a less specific attribute is worth having better accuracy (and

Table 16: Results from chosen classifiers on their base settings and relabeling the class variable to improve metrics

Algorithm	Accuracy	TP.Rate	FP.Rate	Precision	Recall	F.Measure	ROC.Area
ZeroR	73.12%	0.731	0.731	0.000	0.731	0.000	0.500
OneR	76.31% v	0.762	0.540	0.740	0.762	0.792	0.611
J48/C4.5	77.79% v	0.778	0.454	0.761	0.778	0.761	0.718
Naïve Bayes	72.80%	0.731	0.365	0.745	0.731	0.373	0.753
IBk	66.71% *	0.667	0.522	0.664	0.667	0.666	0.573
Simple Logistics	78.55% v	0.786	0.467	0.771	0.786	0.764	0.790
Random Forest	78.72% v	0.789	0.454	0.775	0.789	0.770	0.802

for every other metric, for that matter).

As we already discussed, having a sense of the duration spent in the hospital is still present. Additionally, we can now predict more accurately by a large margin. In contrast, if we come put with a way to produce more sufficient outcomes with the initial dataset, we can always switch back. It seems that the relabeled dataset looks the most promising right now, and therefore we will continue our analysis with this dataset.

When considering classifiers that are best for further use, we consider both Simple Logistics (78.55%) and Random Forest (78.72%) to be most worthwhile. They both scored best across every metric. We will try to optimize these classifiers to increase accuracy even more in the next sections, hopefully.

6.3 Attribute selection

We already discussed an attribute selection regarding the initial dataset, but since we changed the class variable, it is necessary to weigh every attribute again. We consider the same meta learner and evaluators as before; that would be the evaluators CorrelationAttributeEval, InfoGainAttributeEval, and GainRatioAttributeEval. We will discuss the results of each method, as we did earlier.

6.3.1 Correlation

We look at the CorrelationAttributeEval, which calculates the correlation between each attribute and the class variable. When we run an evaluator, we look for values that outline a moderate-to-high or negative correlation, ranging from 1 to -1. We want to avoid attributes with a near-zero or zero correlation, as these attributes can potentially harm our model's performance. Therefore, we want to drop these attributes. The Ranker search method will be used here.

Table 17: Correlation between all independent attributes

Attribute	Score
num_medications	0.35966
num_lab_procedures	0.19294
number_diagnoses	0.18630
num_procedures	0.17575
change	0.08199
number_inpatient	0.06257
hba1c_res	0.05615
A1Cresult	0.05428
insulin	0.05356
diabetesMed	0.04432
readmitted	0.04372
age	0.03356
icu_or_non	0.03096
number_outpatient	0.02507
glyburide	0.02295
gender	0.01707
diag_1	0.01495
metformin	0.01390
glipizide	0.01253
number_emergency	0.00685
pioglitazone	0.00422
race	0.00287
rosiglitazone	0.00155

We observe the results in the table 17, and we see comparable results as seen when performing the evaluator on the initial dataset. At least when talking about the ranking of the attributes. Nonetheless, the scores have improved overall. On the initial dataset, the

best-performing attribute was num_medications with a score of 0.12651 and scores in this setting 0.35966. A cut-off of 0.01 or 0.05 seems doable; 0.01 would eliminate 4 attributes and 0.05 14, respectively. We like to look at the results of other evaluators before making a final decision.

6.3.2 Information gain

Next up is an interesting evaluator called InfoGainAttributeEval, which looks at the information gain or entropy for each attribute for the output/class variable. Calculated values can range between 0 (no information) and 1 (maximum information). Attributes with a high valuation contribute more to the model than lower values do, as they reduce entropy more. There is the potential of removing attributes with a low valuation that could increase accuracy. Like we did in the correlation evaluator, we use the Ranker search method once more. We depict our results and findings in the table 18.

Table 18: Info gain regarding all independent attributes

Attribute	Score
num_medications	0.113531
num_lab_procedures	0.063789
num_diagnoses	0.030539
num_procedures	0.023768
age	0.009319
insulin	0.008625
diag_1	0.006242
change	0.004833
hba1c_res	0.003411
readmitted	0.002796
number_inpatient	0.002619
A1Cresult	0.002614
glipizide	0.001990
glyburide	0.001785
diabetesMed	0.001446
pioglitazone	0.000959
metformin	0.000920
icu_or_non	0.000694
number_outpatient	0.000512
rosiglitazone	0.000503
race	0.000302
gender	0.000210
number_emergency	0.000000

As we have seen in the section about correlation, we see that the same attributes perform worse than others. Some even perform so bad that they get a score of zero - number_emergency - which shows that at least these two attributes are up for removal. Deciding on a cut-off value is difficult; what has considered a 'normal' cut-off value means, again, that we would throw away at least 80% of the attributes. In the case of 0.05, we would be left with only two attributes and with 0.01 eight attributes.

6.3.3 Gain ratio

This evaluator looks a lot like the one we just discussed. It is an enhancement of the InfoGain evaluator, which means it operates with a normalized score. This method measures the significance of attributes concerning the class variable based on the gain ratio in more detail. A higher gain ratio means it reduces more entropy. We again use the Ranker search method. The results are shown in table 19.

Table 19: Gain ratio regarding all independent attributes

Attribute	Gain.Ratio
num_medications	0.06346
num_lab_procedures	0.03877
num_diagnoses	0.02652
num_procedures	0.01968
change	0.01102
insulin	0.00945
diag_1	0.00937
age	0.00707
number_inpatient	0.00671
hba1c_res	0.00671
diabetesMed	0.00640
glipizide	0.00605
glyburide	0.00566
A1Cresult	0.00520
pioglitazone	0.00400
rosiglitazone	0.00363
readmitted	0.00353
icu_or_non	0.00322
metformin	0.00292
gender	0.00147
race	0.01120
number_emergency	0.00000
number_outpatient	0.00000

We observe what we already have seen on the initial dataset and again on this dataset: the same five attributes - num_medications, num_lab_procedures, num_diagnosis, num_procedures, and change - perform the best with each evaluator. The same attributes performed the worst had the same results; number_emergency and number_outpatient had a score of zero. A cut-off point of 0.05 would leave us with one attribute and 0.01 with just the five said best-performing attributes. In the next section, we will discuss, based on each evaluator's results, which attributes are to be dropped to improve our model.

6.3.4 Final decision

As stated and discussed before, we will remove the attributes number_emergency and number_outpatient at least. When deciding on removing more attributes, it becomes a more difficult decision to make. Each evaluator gives variable scores with different sets of criteria, and a one-sided cut-off score is difficult to determine. For example, the insulin attribute scores 0.0521 on correlation, so this would guarantee it to be in the final dataset, as 0.05 is considered a fairly normal cut-off score. However, insulin scores 0.008625 and 0.00945 on information gain and gain ratio, respectively. These different scoring methods make deciding on attributes to be dropped difficult. What is important is that the ranking of attributes is relatable to each other - a low-scoring attribute on the info gain evaluator also scores low with the others. With that, the correlation evaluator can be used for a cut-off score. The scoring here is significantly higher than seen elsewhere - ranging from 0.35 to 0.00155 compared to, for example, 0.06-0.001120 for gain ratio. We are interested in what happens when we perform a cut-off score of 0.01 and 0.05 regarding the correlation evaluator's results. The results are depicted in figure 18.

Table 20: Results from removing certain attributes (new) versus not removing them (old)

Metric	SL.normal	SL.0.01	SL.0.05	RF.normal	RF.0.01	RF.0.05
Accuracy	78.55%	78.59%	77.96%	78.72%	78.45%	74.85%
TP.Rate	0.786	0.786	0.780	0.789	0.785	0.749
FP.Rate	0.467	0.469	0.501	0.454	0.452	0.464
Precision	0.771	0.771	0.764	0.775	0.769	0.732
Recall	0.786	0.786	0.780	0.789	0.785	0.749
F-Measure	0.764	0.764	0.752	0.770	0.766	0.737
ROC-curve	0.790	0.788	0.770	0.802	0.790	0.733

We now observe table 20. When removing below a score of 0.01, we would drop five attributes in total (number_emergency, pioglitazone, race, rosiglitazone, and number_outpatient). Regarding a cut-off score of 0.05, we are removing 14 attributes, and we would be left with a mere eleven attributes. Between the two classifiers, we see different outcomes. When looking at Simple Logistics, we see an increase in accuracy with the cut-off score of 0.01 but a decrease in accuracy when handling a score of 0.05. The increase in accuracy is not very significant; it only accounts for an increase of 0.004%. In contrast, Random Forest does not experience an increase in accuracy with both scores. It drops 0.27% when a 0.01 cut-off is applied and a staggering 3.8% loss with 0.05. A significant loss in accuracy. This concludes that removing the 14 attributes is not worthwhile and, frankly, not in the case of 0.01. Both classifiers do not experience a significant gain in accuracy, and Random Forest is even losing numbers. However, we are still removing the number_emergency and number_outpatient features as these do not reflex our research goals of exploring ways to predict time spent in the hospital.

6.4 Optimalization

In the next section, we want to explore another set of optimization as other classifiers came out best than regarding the initial dataset. The meta learner CVParameterSelection evaluates many iterations to find the best parameter settings with the best-scoring accuracy.

6.4.1 Simple Logistics

Once again, we turn to the classifier Simple Logistics for optimization purposes. When we did this once before, we already discussed all the important parameters and their default settings. We will refrain from researching every one of them once more, as we have seen that some do not greatly impact the performance. Weka also advises using some default settings, for example, with the heuristic stop (H) parameter. We already explored what the effects are if we play with this valuation and saw no reason to change the valuation. Therefore, we only want to explore the maxBoostingIterations (M) and numBoostingIterations (I) parameters again, as we saw promising results from testing on the initial dataset.

Table 21: Results regarding Simple Logistics's base (old) and improved parameter settings (new)

Metric	Old	New
Accuracy	78.59%	78.62%
TP.Rate	0.786	0.786
FP.Rate	0.467	0.467
Precision	0.771	0.772
Recall	0.786	0.786
F-Measure	0.764	0.765
ROC-Curve	0.788	0.789

Looking at the results in table 21, we see a comparison between the outcome gathered from the attribute selection (old) and the best result coming from the CVParameterSelection (new). We tested the I parameter with increasing values by 250, ranging from 0 to 750, and parameter M ranging from 0 to 1.500 in 6 steps. The best outcome was an M of 750 with an I of 0. The accuracy rose by 0.03 percentage points, and the ROC-Curve by 0.01; the rest of the metrics stayed the same. Seeing that we received a better result with these parameter settings, we will base our model on these.

6.4.2 Random Forest

Random Forest is a classifier we have not optimized before. The classifier has many parameters to play with, but we are only interested in one of them: numIterations (I). The (I) parameter is the number of trees generated by the classifier, and the classifier will choose the smallest tree as the best. So the number of trees reflexes the chance of getting the most optimal tree; having a greater number of trees generated can, in theory, give bigger chances of getting a more optimal tree than with lower numbers. However, there is an equilibrium for generating numbers of trees and the chance of getting a more optimal tree. In a moment, there is not going to be a more optimal tree, and more iterations would only increase computing time. We want to explore these boundaries.

Table 22: Results regarding Random Forest's base (old) and improved parameter settings (new)

Metric	Old	New
Accuracy	78.45%	78.77%
TP.Rate	0.785	0.788
FP.Rate	0.452	0.455
Precision	0.769	0.773
Recall	0.785	0.788
F-Measure	0.766	0.768
ROC-Curve	0.790	0.796

We now look at the table 22 and see that the old outcome has a lower accuracy than the new result. We tested with increments of 100, starting from 0 all the way up to 500. The most optimal outcome depicted in the table 22 had a parameter (I) value of 200. Our picked metrics experienced increment; accuracy rose with 0.32% and the precision from 0.769 to 0.773, which were the two biggest risers. Based on these positive results, we will continue with these parameter settings.

6.5 Exploring other meta learners

There are some other ways to improve accuracy, for example, by combining multiple predictions from multiple classifiers. This way of improving accuracy can be done with meta learners that are called ensemble learners. 'Ensemble' meaning a group of items viewed as a whole rather than individually. [6] We already explored Random Forest, an ensemble learner, and can be seen as an extension of Bagging. Besides, we will explore the ensemble learners Bagging, Boosting, Voting, and Stacking in this section.

6.5.1 Bagging

Bagging or Bootstrap Aggregation is used to reduce the variance. As stated before, Random Forest contains bagging elements to do just that. Reducing variance is important, as a high variance means that the model's results can change dramatically by random noise. Thereby making it sensitive to over-fitting. The trick is that each method creates separate samples from the training set and then creates multiple classifiers for each sample. Each classifier gives different results, which gives a good perspective on the performances. Bagging then picks the best-performing classifier.

Since Bagging already occurs in the Random Forest classifier, we will not include it here. This would be counterproductive. The results depicted in table 23 are from Simple Logistics only.

Table 23: Results regarding meta-learner Bagging on Simple Logistics (new) and without its influence (old)

Metric	Old	New
Accuracy	78.62%	78.62%
TP.Rate	0.786	0.786
FP.Rate	0.467	0.467
Precision	0.772	0.772
Recall	0.786	0.786
F-Measure	0.765	0.765
ROC-curve	0.789	0.789

Looking at the results depicted in table 23, we can see the most-optimal model (old) and the results of the bagging classifier (new). Bagging did not perform better than the older model. The reason could be that there is not much variance in our model; therefore, bagging could not improve accuracy. Based on the results, we will take the older model and retire the new one.

6.5.2 Boosting

Boosting classifiers are mostly used to reduce bias in a model. There are multiple algorithms available, such as AdaBoost and Decision Trees. We are going to use AdaBoost in this section as it is the most accepted meta-classifier for Boosting. A high bias can cause under-fitting by missing out on information and correlations between the class and independent variables. It does this by starting with a base classifier prepared on the training set. After that, a second classifier is initiated to focus on wrongly predicted instances by the first run. This process continues with adding classifiers until the accuracy becomes stable or the limit of models is reached.

Because Random Forest is already a meta learner, we will not include it in this section. [7]

Table 24: Results regarding meta-learner Boosting on Simple Logistics (new) and without its influence (old)

Metric	Old	New
Accuracy	78.62%	78.62%
TP.Rate	0.786	0.786
FP.Rate	0.467	0.467
Precision	0.772	0.772
Recall	0.786	0.786
F-Measure	0.765	0.765
ROC-curve	0.789	0.789

We observe the table 24, where the results of the AdaBoost are depicted. We see a similar result as seen with the Bagging classifier; the accuracy does not improve across these meta-learners. AdaBoost may seem not to be able to get the accuracy of a stronger learner up. It may be that AdaBoost seems more appropriate for a learner with weak results ($<= 1/2$ accuracy). Therefore, we will not go further with the model created by AdaBoost.

6.5.3 Voting

Voting is perhaps the most simple ensemble learners but is still fairly effective. It works by selecting two or more classifiers and selecting what criteria all classifiers will be combined. The default setting for the combination rule is taking the average of probabilities, but we will use Majority Voting. When all classifiers are combined, they can vote on what the outcome should be.

Table 25: Results regarding meta-learner Vote on Simple Logistics and Random Forest (new) and without its influence (old)

Metric	ZeroR	OneR	J48	Naïve.Bayes	Simple.Logistics	IBk	Random.Forest	Vote
Accuracy	73.12%	76.31%	77.79%	72.80%	78.62%	66.71%	78.77%	78.66%
TP.Rate	0.731	0.762	0.778	0.731	0.786	0.667	0.788	0.787
FP.Rate	0.731	0.540	0.454	0.363	0.467	0.522	0.455	0.498
Precision	0	0.740	0.761	0.745	0.772	0.664	0.773	0.776
Recall	0.731	0.762	0.778	0.731	0.786	0.667	0.788	0.787
F-Measure	0	0.792	0.761	0.733	0.765	0.666	0.768	0.757
ROC-curve	0.500	0.611	0.718	0.753	0.789	0.573	0.796	0.644

Looking at the table 25, we can see the best results retrieved from the performance of the Vote classifier. We tested on every used classifier, not regarding their past performances. The outcomes-based on Simple Logistics and Random Forest have been adjusted to their optimal parameter settings, as discussed before. We see that the Vote classifier produces relatively high accuracy, precision, and recall. It performed as the runner-up after Random Forest, with only a difference of 0.11% in accuracy. The FP rate of the Vote is quite higher than that of Simple Logistics, which performed third-best. Looking at the results, we can conclude that the Random Forest algorithm is the biggest contender. For that reason, we will not include the Vote as a potential candidate for the final model.

6.5.4 Stacking

Stacking looks like Voting as both works with multiple classifiers and their results. However, there is a huge difference. Stacking does not 'vote' on which classifier is the best-performing; it contains a meta-learner that learns how to combine the predictions from sub-models best. It can therefore increase our accuracy. We will test the stacking classifier on both Random Forest and Simple Logistics with Logistics as the meta-learner, as this classifier is widely used. Again, we test with the most optimal parameter settings.

Table 26: Results regarding meta-learner Stacking on Simple Logistics and Random Forest (new) and without its influence (old)

Metric	Simple.Logistics	Random.Forest	Stacking
Accuracy	78.62%	78.77%	79.04%
TP.Rate	0.786	0.788	0.790
FP.Rate	0.467	0.455	0.442
Precision	0.772	0.773	0.777
Recall	0.786	0.788	0.790
F-Measure	0.765	0.768	0.773
ROC-curve	0.789	0.796	0.803

We now observe table 26, where we see that Stacking actually increased the accuracy to a new highest level. It did this to 79.04%, outperforming both Simple Logistics and Random Forest. The difference in performance between all models is not significant, but a better and more accurate model is always welcomed. We, therefore, will take the Stacking outcome to the next section as potentially our final model.

6.6 Confusion matrices

This section looks at the confusion matrices of the best-fitting versions of our chosen classifiers, which are Random Forest and the combining of classifiers in the Stacking meta-learner. Since we have a class variable with only 2 values, a matrix would have a 2x2 structure. Now that we want to make a final decision on which model we are going to use, with the fact that all our models' performances are relatively close to one another, it can help to look at the details for making a final decision.

6.6.1 Random Forest

First, we are going to look at the table 27. Here, the confusion matrix retrieved from the best-performing outcome of Random Forest is depicted. Random Forest scored an accuracy of 78.77% at its most optimal. The matrix is based upon these results.

a	b	<– classified as
47830	3672	a = 1
11283	7653	b = 2

Table 27: Confusion matrix based on the optimal settings regarding Random Forest

We observe the results from the matrix in the table 27. The data is still positively skewed as 5.9113×10^4 instances are classified as 'a' and 1.1325×10^4 as 'b'. There seems to be a difference in metrics scores between classes. The b class scores less than class a—for example, the TP.Rate for a is set at 0,929 and b at 0,404, which means that there is still a bias for class a. In contrast, the correctly predicted instances have been upped significantly, and we can look at the differences between the classes more precisely. It is still possible to determine what needs to change to reduce hospital visits' duration to a much better degree than the initial model.

6.6.2 Stacking

Next up is the Stacking classifier, which scored the best metrics scores seen thus far. We achieve this by combining the two best-performing classifiers, Random Forest, and Simple Logistics. The scored accuracy is set at 79.04%. The accompanying confusion matrix is depicted in the table 28.

a	b	<– classified as
47669	3833	a = 1
10928	8008	b = 2

Table 28: Confusion matrix based on the combining of Random Forest and Simple Logistics with the meta-learner Stacking

Looking at the results, we see a picture complementary to that of Random Forest. The number of positives is bigger with a tiny margin (with 194 instances, to be exact). The difference comes from the fact that there are less true positives but more true negatives than Random Forest. Looking at the distribution of predicted instances, we can decide the classifier that will wrap our model.

6.6.3 Conclusion

Now that we have a better picture of the distribution of predictions, we can decide on a final model to base our findings on. As already discussed, the accuracy and other Stacking's metrics looked a bit better than Random Forest alone. Therefore, it was not totally necessary to discuss both confusion matrices, as it was most likely to take Stacking as the final model. Nevertheless, looking at both matrices, we can determine where the differences are; although Stacking predicts less correct on class 1 (47.830 vs. 47.669 instances), it does relatively better predict class 2 than Random Forest (7.653 versus 8.008 instances). This means that the total correctly predicted instances (TN + TP) is higher with Stacking than observed in Random Forest. For that reason, Stacking will be appointed as our final model.

6.7 A ROC curve

The last thing we want to look at before deciding on a final model is the receiver operating characteristic (ROC) curve. This curve represents a fraction between the true positive rate and the false positive rate. The curve is essentially a representation of a confusion matrix. A ROC space is created by plotting the FPR and TPR as x and y axes to depict their relationship. A diagonal line divides this space; a point above this line represent good results from a classifier (high TPR, low FPR), whereas points below the line present a not so good result (low TPR, high FPR). In this section, we take a look at the ROC curve of Stacking and analyze its results.

```
roc_data <- read.table(
  "datasets/dataset_results_weka/roc_curve/roc-curve_stacking.arff",
                    sep = ",",
                    comment.char = "@")
names(roc_data) <- c("Instance_number",
                  "True_Positives",
                  "False_Negatives",
                  "False_Positives",
                  "True_Negatives",
                  "False_Positive_Rate",
                  "True_Positive_Rate",
                  "Precision",
                  "Recall",
                  "Fallout",
                  "FMeasure",
                  "Sample_Size",
                  "Lift",
                  "Threshold")

library(ggpubr)
colors <- c(classifier = "red", threshold = "blue")
plt <- ggplot(data = roc_data,
       mapping = aes(x = False_Positive_Rate, y = True_Positive_Rate)) +
    geom_point(mapping = aes(color = "classifier")) +
    geom_abline(aes(color = "threshold",
                    slope = 1,
                    intercept = 0)) +
    scale_color_manual(values = colors) +
    xlab("True Positive Rate") +
    ylab("False Positive Rate") +
    theme_pubr() +
    theme(legend.title = element_blank())
```

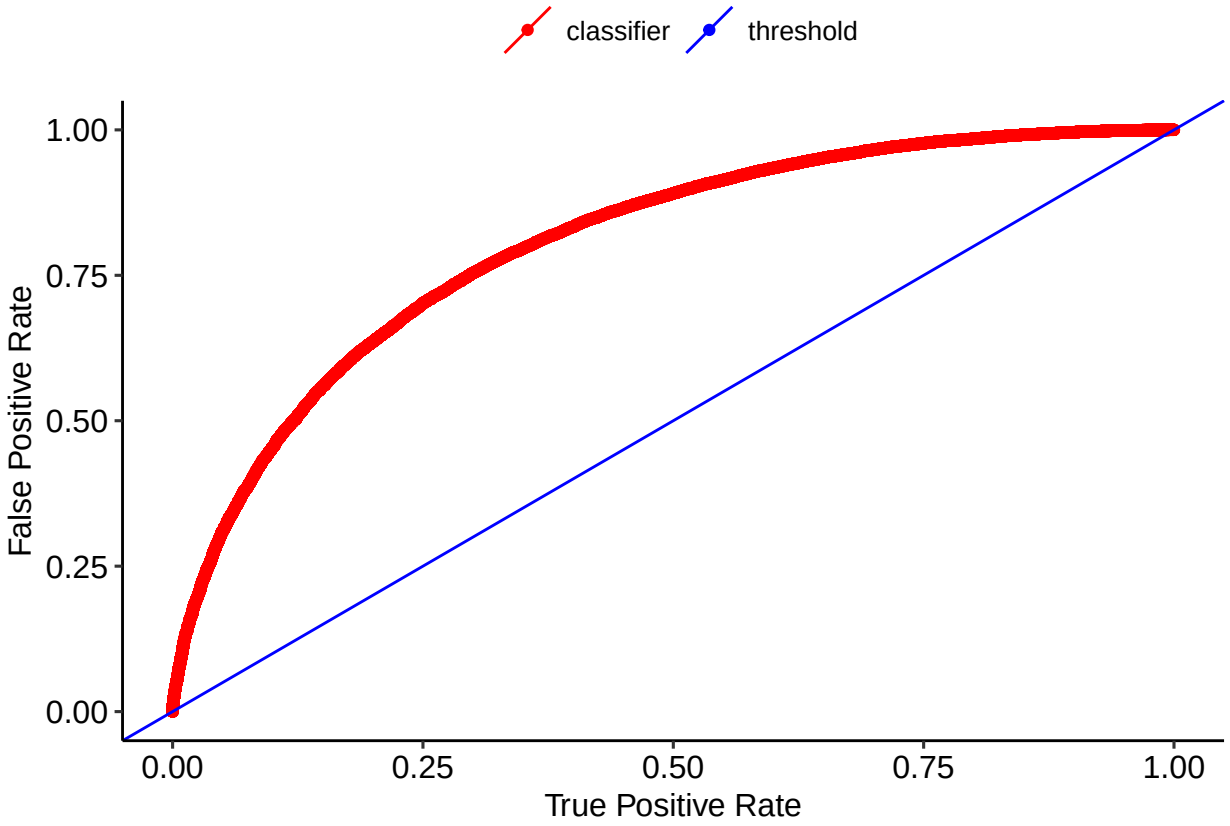


Figure 25: ROC Curve based on the combining of Random Forest and Simple Logistics with the meta-learner Stacking

Looking at figure 23, we see the ROC curve of stacking depicted. The red line is the classifier's performance, and the blue line divides the ROC space. We notice that our classifier performs well above the diagonal line, which means the classification is good. The optimum point seems to be around a TPR of 0.65 and an FPR of 0.25. This means that the positive cases that are predicted are actually positive ones, indicating high recall. A high recall/sensitivity is something we strive for. The Area under the ROC, which also one of our chosen metrics, is set at 0.803 and can be used as an accurate indicator. A high ROC Area means good accuracy. Our final model scores descent enough values to be a good model.

Our next goal is to create a JavaWrapper that can accept new instances and analyze these based on our model. The wrapper can be found in this BitBucket repository.

7 References

References

[1] Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore, Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records, BioMed Research International, vol. 2014, Article ID 781670, 11 pages, 2014. DOI: https://doi.org/10.1155/2014/781670

[2] Centers for Disease Control and Prevention, National Center for Health Statistics, ICD-9, https://www.cdc.gov/nchs/icd/icd9.htm, November 6, 2015.

[3] Stiglic, G., Kocbek S., Pernek I., Kokol P., Comprehensive Decision Tree Models in Bioinformatics. DOI: 10.1371/journal.pone.0033812

[4] Eibe Frank, Administrator; Logistic VS Simple Logistic, https://weka.8497.n7.nabble.com/Logistic-VS-Simple-Logistic-td31410.html

[5] Hall, M, Class AttributeSelectedClassifier, https://weka.sourceforge.io/doc.dev/weka/classifiers/meta/AttributeSelectedClassifier.html.

[6] Thornton, C.; Hutter, F.; Hoos, H.H.; Leyton-Brown, K.: Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms, https://arxiv.org/pdf/1208.3719.pdf.

[7] Boinee, P; De Angelis, A; and Foresti, G.L.: Meta Random Forests, International Journal of Computational Intelligence Volume 2 Number 3, https://www.researchgate.net/publication/242416336_Meta_Random_Forests.