CYO Project - Efficacy of Several Popular ML Algorithms

KST

11-June-2025

Contents

Acknowledgements

As earlier, I would like to thank Professor Rafael A Irizzary , The Teaching & Support Staff of the Department of Biostatistics, The Harvard University and the edX platform for providing students from all over world the opportunity to learn the fundamentals of Data Science at their own pace and convenience. The availability of such facilities and the accommodating structure of the course is valuable beyond words to those who are disadvantaged for time and who otherwise cannot afford to attend classroom sessions or learn according to a fixed schedule.

There are several references listed towards the end of the this document and I would like to express my heartfelt gratitude to the Authors, Contributors and Collaborators of the articles that they refer to.

I would also like to thank the Authors, Contributors and Collaborators of the articles available at the URL listed below, for the guidance provided by these articles in the execution of the project and the preparation of this report.

https://en.wikipedia.org/wiki/International_Classification_of_Diseases

https://en.wikipedia.org/wiki/List_of_ICD-9_codes

https://keras3.posit.co/

https://tensorflow.rstudio.com/

https://stringr.tidyverse.org/

https://stackoverflow.com/questions/50820409/sorting-output-from-variable-importance-fcaret-package

https://www.r-bloggers.com/2018/12/rstudio-pandoc-html-to-markdown/

https://rmarkdown.rstudio.com/docs/reference/pandoc_convert.html

https://dplyr.tidyverse.org/reference/mutate_all.html

https://www.r-bloggers.com/2023/01/imputation-in-r-top-3-ways-for-imputing-missing-data/

https://datatricks.co.uk/one-hot-encoding-in-r-three-simple-methods

https://cran.r-project.org/web/views/MachineLearning.html

https://www.efta.int/sites/default/files/R2022.pdf

https://www.ebpi.uzh.ch/dam/jcr:dc0cef17-29c7-4e61-8d33-e690561ab7ae/mi_intro20191001.pdf

https://bookdown.org/yihui/rmarkdown/markdown-syntax.html

https://bookdown.org/yihui/bookdown/tables.html

https://bookdown.org/yihui/rmarkdown-cookbook/purl.html

*Note: All referenced URLs in this report were current and the content accessible at the time of writing this report.

1 Introduction

1.1 Project Background

As our awareness grows about the multitude of Machine Learning (ML) Algorithms available for Data Analysis and the kind of problems they solve, one of the most important questions that arises is, which of them works the best in any given situation? Each of these Algorithms have their own strengths and weaknesses. A lot of the time, the datasets and the analysis tasks at hand themselves have a large role to play in the efficacy of these Algorithms. Even if the datasets address similar topics, the structure and contents of the datasets vary widely.

Through the course of this Project we will investigate the utility of several popular and well respected ML Algorithms as applied to a few publicly available datasets to understand what kind of performance and efficiency we can expect from them.

1.1.1 Project Success Criteria

There are no defined quantitative Project Success Criteria as such. Success is assessed qualitatively based on the the insight we can gain from the project and if it will help us foster our learning about these Algorithms and increase our awareness of their applicability in different situations.

Within this context, our Project can be seen as being successful if Readers, when handed a new dataset to analyse, can use the information from the Project to answer questions like :-

1. How to assess the dataset and the associated complexity
2. How to get started with analysing the data
3. What kind of processing might be required
4. Which Algorithms to use to get some initial insight
5. Which ones to use for more detailed analysis.

1.2 Project Approach

The first task in starting off with the Project is to obtain publicly available and unencumbered datasets that can help satisfy the eligibility criteria for the Project to be considered authentic, insightful and innovative while not being a repetition or modification of work that is already done. The University of California, Irvine (UCI) Machine Learning datasets (https://archive.ics.uci.edu/datasets) in particular have been worked upon by a multitude of students from those at the Undergraduate level to those who are submitting their Doctoral thesis. Even a cursory Internet search throws up several pages of independent projects for each of the popular datasets. Needless to say, the first task is itself quite daunting.

As a part of this project we will try to build something that is useful practically. We will also try to choose datasets that are still not as frequently used as the others.

As a common theme, we will choose datasets that are related to Cardiovascular Diseases (CVD). As anybody with personal experience can tell, CVD can have a huge bearing on the quality of life of those afflicted and their caregivers.

As a quick reminder of the debilitating nature of CVD, here is what the World Health Organisation (WHO) has to say about them.

Cardiovascular diseases (CVDs)

Cardiovascular diseases (CVDs) are the leading cause of death globally, taking an estimated 17.9 million lives each year. CVDs are a group of disorders of the heart and blood vessels and include coronary heart disease, cerebrovascular disease, rheumatic heart disease and other conditions. More than four out of five CVD deaths are due to heart attacks and strokes, and one third of these deaths occur prematurely in people under 70 years of age.

More Information is available about CVD at https://www.who.int/health-topics/cardiovascular-diseases#tab=tab_1

1.2.1 Choice of Datasets

Our choices for the datasets, in increasing order of complexity and difficulty of prediction of the outcome(s) are:-

1. Heart_Failure_Prediction (HFP) - The dataset is available at

   https://www.openml.org/search?type=data&status=active&id=45950&sort=runs.

   This is a relatively newer dataset. It is very simple as well

   The dataset uses the "Attribute-Relation File Format".

2. Myocardial infarction complications (MIC) - The dataset is available at https://archive.ics.uci.edu/dataset/579/myocardial+infarction+complications.

   The DOI in the dataset description points to the availability of the Comma Separated Variable (CSV) version of the dataset hosted at the University of Leicester's repository at

   https://figshare.le.ac.uk/articles/dataset/Myocardial_infarction_complications_Database/12045261/3.

   We will use the CSV version available at the University of Leicester's repository as it is easier to process. The CSV file is already formatted with the column headers and unknown values are represented using "?" marks.

3. Diabetes 130-US Hospitals for Years 1999-2008 (d130) - The dataset is available at

   https://archive.ics.uci.edu/dataset/296/diabetes+130-us+hospitals+for+years+1999-2008.

   This dataset is more dated having been donated in 2014. Needless to say, there are several volumes of Projects based on this dataset. The list of Attributes/Features is very well described in the Research Paper available at

   https://doi.org/10.1155/2014/781670.

   The dataset is available in CSV format.

Through all 3 datasets, we will also explore the utility of Artificial Neural Networks and their suitability for the tasks at hand.

1.2.2 Dataset partitioning scheme for Training, Validation and Testing

We will create Training and Testing(Holdout sets) in the ratio of 80:20.

We will further split the Training set into "Training CV" and "Testing CV" sets in the ratio of 80:20. We will use "Training CV" for training our Algorithms and "Testing CV" for validation. For those who have not guessed it already, CV stands for Cross Validation.

We will follow this split or partitioning scheme for all of our datasets.

The choice of 80:20 is based on the size of the datasets which are quite small. With 80:20, we have the ability to gauge the performance of the algorithms on "Testing" data that is likely to vary from the "Training" data simulating actual performance.

With a split like 90:10, we would likely end up with too little "Testing" data to simulate actual performance.

With a split like 70:30 or 60:40, we would likely end up with too little "Training" data.

1.2.3 Project Report

For the Project report, We will organise the Analysis and Results for each dataset into a separate chapter. This will help prevent overcrowding of information into any single chapter and also make it easier to navigate and follow the report.

To keep the length of the Report manageable and to avoid confusion and needless repetition. I have included only the report text and the outputs in the PDF format of the Report while the Rmd format contains the whole report text and code. As might be expected, the R file contains the entire code.

1.3 Additional Notes and Cautions

Here we record some Notes and Cautions that are related to this Project. This might help the Readers in understanding the Report better and also help them avoid some of the problems that I have faced and spent much time recovering from.

Artificial Neural Networks (ANN)

Artificial Neural Networks(ANN) are today some of the most sought after Machine Learning Techniques. They offer a lot of flexibility in building Algorithms due to their programmable nature. There are many ANN packages available. Some are pretty basic and used for simple tasks while some are extremely advanced and used for building some of the most complex Large Language Models (LLM).

A summary of the various Machine Learning packages, including ANN, that are available for the R language and their relative merits and demerits is provided at :-

https://cran.r-project.org/web/views/MachineLearning.html

Just like with the rest of our Project, our ambitions with ANN will be very grounded. ANN can be built to be extremely sophisticated and accurate. Building such ANN require plenty of time, effort and expertise. For this Project, our main aim will be to introduce ANN as a choice for Machine Learning and demonstrate how they can be built for achieving different goals. It is meant for those who are just starting off with ANN.

For building ANN, we will use Keras and particularly Keras 3 which is the current major revision of Keras. This report has been prepared with "keras3 version 1.3.0" released 2025-03-03.

Keras is well documented, easy to understand and use, particularly so for beginners. However, it is not a native R package in entirety. Keras is a deep learning API written in Python and capable of running on top of back-end libraries like JAX, TensorFlow or PyTorch.

The Keras R Package provides some wrapper functions that format the code written in R into a format understood by TensorFlow or JAX. All computation is done in Python and the results returned to R. This requires the installation of a Python Virtual Environment and the required Python libraries into the same.

The Procedure is fairly involved, needs large downloads, and is only recommended for those who are interested in running the code to check if it works as demonstrated.

Also care should be taken to ensure that only Keras 3 packages for R are installed and not the older Keras 2 packages as they could lead to conflicts and inexplicable behaviour with no explicitly displayed warnings about the version mismatches. For historical reasons CRAN refers to Keras 2 as Keras.

If Graphical Processing Unit (GPU) usage for computations is desired, the installation is even more complex as the requisite packages for GPU computing need to be installed on the Operating System as well as within the Python Virtual Environment created for Keras. I have deliberately disabled usage of GPU computing in the environment to avoid any additional complexity. Most of the ANN are extremely simple and should run on any modern Laptop computer in quick time.

The other behaviour to watch out for in Keras is that the results are not always consistent unless the Random Number Generator (RNG) seed is set for multiple different processing entities. Even with the RNG seed set, the models often provide different predictions between runs or when multiple models are run one after the other. This behaviour is observed even with all parameters being unchanged and using the CPU (instead of the GPU) for computations. After numerous failures and plenty of time lost, I have used a rather inelegant hack to get it to provide consistent results. However, the hack by itself is not sufficient when tuning the ANN, it is necessary to frequently check that the ANN works as intended by restarting R and running it afresh. Though frustrating during the tuning stage, it saves a lot of time and heartache in the longer run.

You can read more about Keras and Keras for R at the following Internet URL.

https://www.keras.io

https://keras.posit.co

Keras provides 2 main API varieties, Sequential API and Functional API with differing complexities and capabilities. More information is available at the URL mentioned.

Using Different Backend Frameworks with Keras

You can read more about TensorFlow and JAX at the following URL:-

https://tensorflow.rstudio.com

https://www.tensorflow.org/learn

https://docs.jax.dev/en/latest/beginner_guide.html#beginner-guide

I have used JAX as the Back-end, if the readers and evaluators decide to choose a different Back-end, the results might be slightly different.

Tuning Keras

Tuning Keras requires a decent understanding of the API, various parameters and how they affect predictions. Some of the parameters that have been tuned in this report are :-

a. Sequential API
    1. Types of Layers
    2. Numbers of Layers
    3. Number of Units in each Layer
    4. Activation Functions
    5. Dropouts
    6. Batch Size
    7. Number of Epochs
    8. Class Weights
b. Functional API
    1. All parameters in Sequential API
    2. Merging Layers

All of these parameters work in concert to affect the predictions.

While changes due to batch size or the number of epochs are easy to track and visualise, Even small changes in parameters like the number of layers, number of units in a layer, dropout rates and the RNG seed used for dropout can sometimes vary the predictions drastically.

For tuning, the creators of Keras recommend using the Keras Tuner which is natively built in Python.

https://keras.io/keras_tuner/

An R wrapper is available,

https://eagerai.github.io/kerastuneR/

I have not used the Keras Tuner and have manually tuned the ANN in this report. The reasons are twofold.

- Gain some insight into how each parameter affects the results and by how much.
- Avoid any additional complexity in building the ANN.

The Readers can definitely explore using the Keras Tuner as they deem fit.

Printing Keras Models

I have disabled printing of the Models for ANN in the Rmd File and the R file. The reason is twofold.

1. One needs to have an understanding of Keras API to understand what they mean.
2. Reading them is not easy, they can be quite long and not very intelligible.

Readers who are interested in viewing them can uncomment the lines and have a look. I have provided the full code in both the Rmd and R files. The code provides greater insight into the way the Model is built rather than the Model Prints.

Imputation

Many packages are available for imputation and some of them like Visualization and Imputation of Missing Values (VIM)and Multivariate Imputation by Chained Equations (mice) are extremely capable and good at imputing values. The choice of the packages is largely dependent on the dataset and the missing values and the algorithms that can be used to impute the missing values. Not to restate what is well articulated already, please refer to the following URL for more information about the topic.

https://www.efta.int/sites/default/files/R2022.pdf

https://www.ebpi.uzh.ch/dam/jcr:dc0cef17-29c7-4e61-8d33-e690561ab7ae/mi_intro20191001.pdf

For some of our datasets, particular the MIC dataset, imputation is an absolute necessity to run algorithms like ANN and Random Forest as these algorithms cannot work with missing values.

The accuracy of these algorithms is also largely dependent on imputation and any imputation that introduces inaccuracies into the dataset, also causes the predictions from the algorithms to be inaccurate.

ANN particularly suffer a lot from inaccurately imputed values.

This inaccuracy manifests itself in both Training and Testing. In Training, it can lead to inappropriate tuning of the ANN. In Testing, it often leads to results showing a poorly performing ANN because the Testing data has inaccurately imputed values for the features.

The imputation packages also offer a lot of tuning capabilities that can be used to increase the accuracy of the imputations. I have used capabilities that can be run in sufficient time and with sufficiently capable computing resources.

R Markdown Conversion to PDF

As indicated in the sibling report for the "MovieLens" project, R Markdown conversion to PDF is not very straightforward.

This particularly affects the column widths of tables and is quite well known within the LaTeX user community.

Tables generated using R code can be handled a lot better using the "chunk" options, the "kable" and "kableExtra" packages.

For more details, please refer to

https://bookdown.org/yihui/rmarkdown/markdown-syntax.html

https://bookdown.org/yihui/bookdown/tables.html

Being able to reproduce the content in the PDF document as closely as possible to the Rmd document requires good knowledge of LaTeX. I have done what I could to keep the documents as close as possible. When in doubt, please consider the Rmd document to be the single and most authoritative source of truth.

Usage of Forward Pipe Operator (|>) and Tidyverse magrittr Pipe Operator (%>%)

Both types of Pipe Operators are used in the code.

The Forward Pipe Operator(|>) is available in the R Base package. It is used in places where it should make it more convenient for those who wish to compare the code with ANN examples from other sources.

The Tidyverse magrittr Pipe Operator (%>%) is available from the Tidyverse package and has been already used extensively in the courses in the "Data Science Series" and should be convenient for anybody who wishes to follow or compare the data processing and transformation operations using Tidyverse.

XGBoost

The versions of XGBoost available on CRAN at https://cran.r-project.org are quite dated. The newer packages are available at https://dmlc.r-universe.dev and https://cloud.r-project.org.

The newer packages use slightly different syntax which are more friendly and are also more accurate.

Support for "categoricals" is available in R through factors from XGBoost version 3.0 onwards, however the feature is considered experimental. According to the developers, only the Python package is fully supported.

https://xgboost.readthedocs.io/en/stable/tutorials/categorical.html

I have not used "categoricals" in this report to avoid any additional complexity.

This report has been prepared with "xgboost version 3.0.0.1" released 2025-03-14.

If readers are already using an older version of XGBoost, they may see some differences in the values reported.

Reuse of Variable names

Compared to the sibling project, where I made a concious effort to keep the names of variables used for intermediate processing and reporting results completely separate, in this project, I have reused the variable names used for intermediate processing and reporting results as the code has a lot of similarities. Using the same variable names makes it easier to build, reproduce, debug and follow the code. However for this to work the variables need to be sanitised or deleted before reusing their names again.

As an example, almost all of the ANN follow a similar pattern and also use the same variable names making it a lot easier to compare different ANN.

In some places where I need to compile the variable names for reporting at the end, I have made a conscious effort to make the variable names self describing. Sometimes this has lead to extraordinarily long variable names.

One-Hot Encoding

For one-hot encoding, either the "one_hot" function from the "mltools" package can be used or a combination of the "dummyVars" function from the "caret" package and the generic predict function from the R "stats" package can be used.

Using the one_hot function is simple and the code is easier to follow. However, one caveat is that one_hot only supports unordered factors. dummyVars supports both ordered and unordered factors. dummyVars has a slightly more complex interface that needs a little bit of additional effort to follow.

I have used both one_hot and dummyVars in the Project based on need and simplicity.

There are a few places in the MIC dataset, where I have used two instances of dummyVars, one for the predictors and one for the outcome. The only reason for doing it was simplicity over brevity or concise code. I had initially built the code with only a single instance of "dummyVars" but soon realised that it would be too confusing for the readers to follow along. The biggest drawback in using a single instance is that the outcome could easily leak into the predictors. Keeping them separate avoids any such leakages.

ConfusionMatrices

I have used a lot of confusion matrices to depict the results, though they appear overwhelming or tedious sometimes, they are an Analyst's best friend when categorical outcomes are involved. At a glance, they

can provide a lot of very important information about metrics like Overall Accuracy, Balanced Accuracy, Sensitivity, Specificity, The number of accurate and inaccurate predictions organised very conveniently. They help us easily gauge the performance of our Algorithms.

2 Heart Failure Prediction (HFP)

The first dataset that we will pick up is the Heart_Failure_Prediction(HFP) dataset. The dataset is extremely simple, well structured and easy to work with..

2.1 Dataset Introduction

The dataset, consists of 5000 clinical records of patients with Heart Failure. It includes the attributes described below.

Table 1: Heart Failure Prediction - Attributes/Features

Col Id	Name	Description	Type	Values
1	age	age of the patient	Numeric	48 Distinct
2	anaemia	presence of anaemia	Binary	0: No 1: Yes
3	creatinine_phospokinase	level of creatinine phospokinase in the blood	Numeric	290 Distinct
4	diabetes	presence of diabetes	Binary	0: No 1: Yes
5	ejection_fraction	percentage of blood leaving the heart at each contraction	Numeric	17 Distinct
6	high_blood_pressure	presence of high blood pressure	Binary	0: No 1: Yes
7	platelets	platelet count in the blood	Numeric	203 Distinct
8	serum_creatinine	level of serum creatinine in the blood	Numeric	43 Distinct
9	serum_sodium	level of serum sodium in the blood	Numeric	27 Distinct
10	sex	gender of the patient	Binary	0: Female 1: Male
11	smoking	smoking status of the patient	Binary	0: No 1: Yes
12	time	follow-up time	Numeric	155 Distinct
13	DEATH_EVENT	indicator of death occurrence during the follow-up period	Binary	0: No 1: Yes

Exact units of the attributes are not provided but it is quite easy to decipher some of them like age (years) and time (days) from the range of values. The actual units are not relevant to our predictions.

***Note: For this dataset, we will treat Numeric variables as being Continuous**

This dataset can be used for analysing the relationship between various clinical attributes and the occurrence of death events in patients with heart failure. It can help in predicting the risk factors associated with heart failure mortality.

2.2 Analysis

As indicated in the Introduction, the dataset is available from the https://openml.org website and is provided in the "Attribute-Relation File Format" (ARF) format. The data can easily be extracted into a data frame for further processing. Let us have a look at the structure of the dataset. We will also record the survival rate of patients for reference.

```
## [1] Structure of the HFP dataset

## 'data.frame':    5000 obs. of  13 variables:
##  $ age                     : num  55 65 45 60 95 70 63 70 50 53 ...
##  $ anaemia                 : num  0 0 0 1 1 0 1 1 0 1 ...
##  $ creatinine_phosphokinase: num  748 56 582 754 582 232 122 171 482 446 ...
##  $ diabetes                : num  0 0 1 1 0 1 1 0 1 0 ...
##  $ ejection_fraction       : num  45 25 38 40 30 30 60 50 30 45 ...
##  $ high_blood_pressure     : num  0 0 0 1 0 0 0 1 0 1 ...
##  $ platelets               : num  263358 305000 319000 328000 461000 ...
##  $ serum_creatinine        : num  1.3 5 0.9 1.2 2 1.2 1.2 0.9 0.9 1 ...
##  $ serum_sodium            : num  137 130 140 126 132 132 145 141 132 133 ...
##  $ sex                     : num  1 1 0 1 1 1 0 0 1 1 ...
##  $ smoking                 : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ time                    : num  88 207 244 90 50 210 147 196 109 215 ...
##  $ DEATH_EVENT             : num  0 0 0 0 1 0 0 0 0 0 ...

## [1] The Survival Rate of patients in the HFP dataset is :
## [2] 0.6864
```

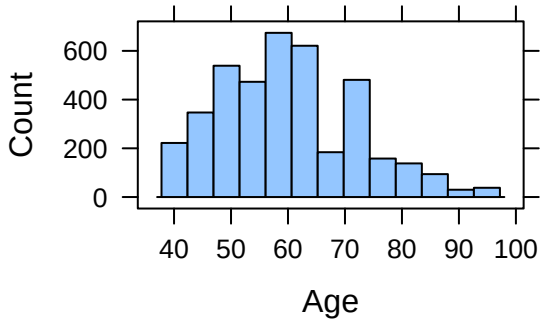
2.2.1 Dataset Preparation

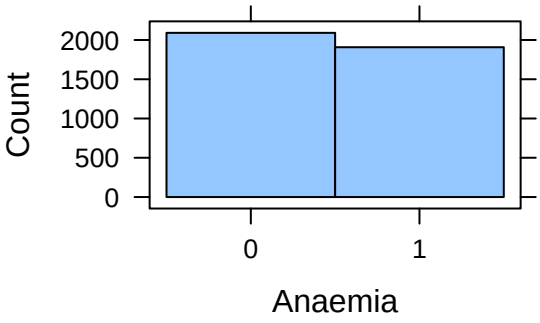
Very little preparation if any is required to work with this compact and extremely well structured dataset. The only dataset modification required is to convert the binary variables and the outcome as "factors". We will treat the rest of the variables as continuous variables.

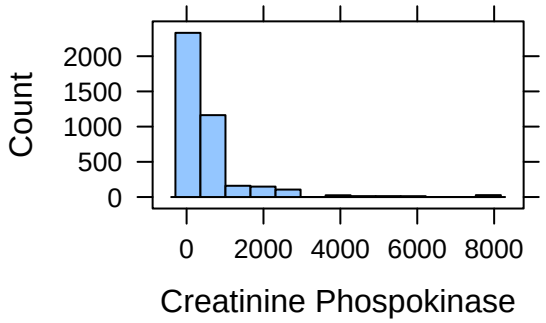
We will then create our Training, Testing, Training CV and Testing CV datasets.

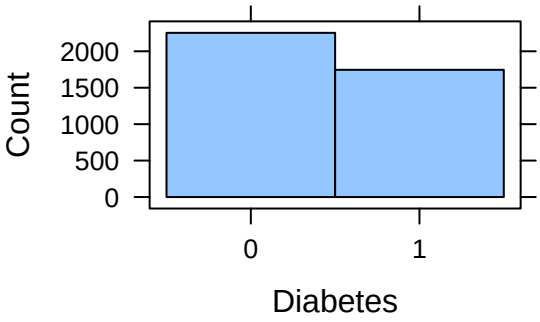
Distribution of Variables

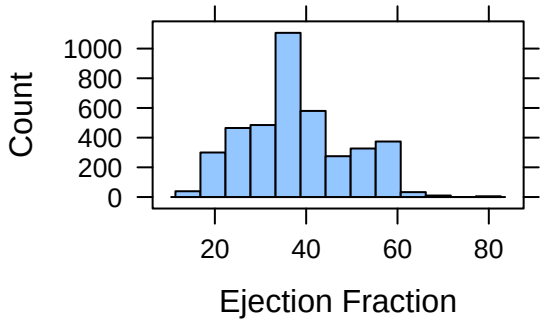
Let us visualise the distribution of the variables. We will only look at the values for the "Training" dataset.

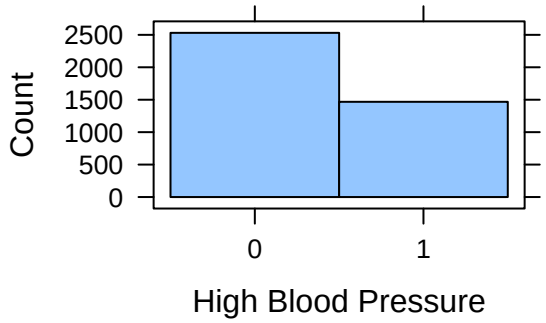

Count
600
400
200
0
40
50
60
70
80
90
100
Age

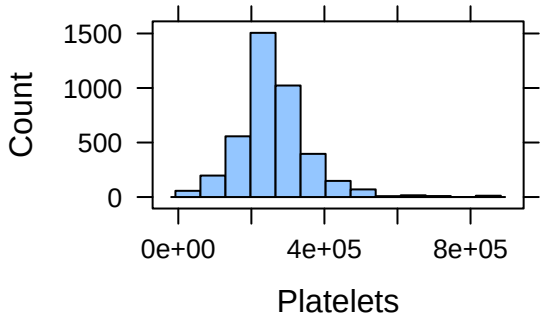

Count
2000
1500
1000
500
0
0
1
Anaemia

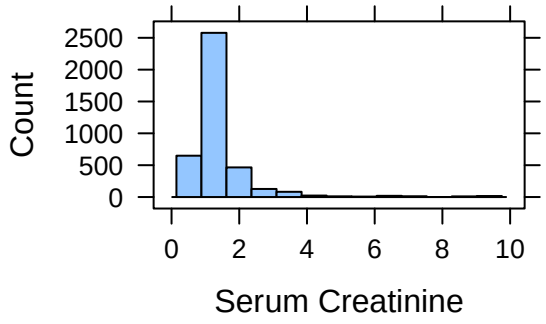

Count
2000
1500
1000
500
0
0
2000
4000
6000
8000
Creatinine Phospokinase

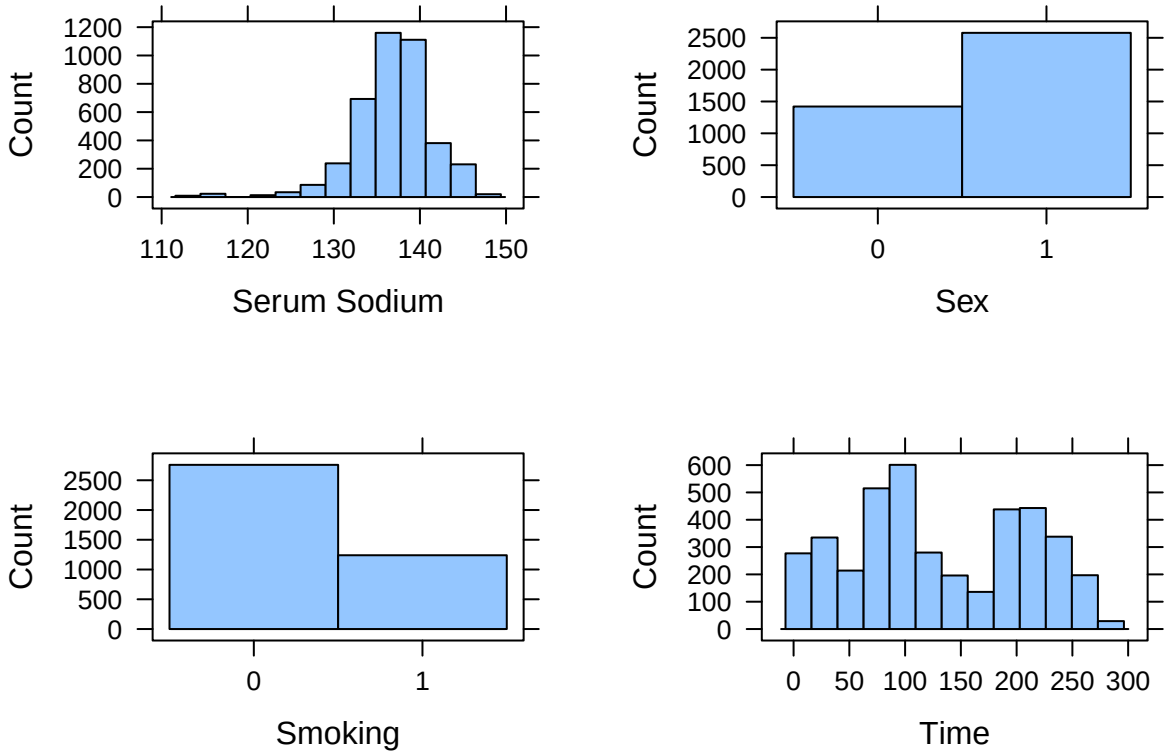

Count
2000
1500
1000
500
0
0
1
Diabetes


Count
1000
800
600
400
200
0
20
40
60
80
Ejection Fraction


Count
2500
2000
1500
1000
500
0
0
1
High Blood Pressure


Count
1500
1000
500
0
0e+00
4e+05
8e+05
Platelets


Count
2500
2000
1500
1000
500
0
0
2
4
6
8
10
Serum Creatinine



We will start with some basic algorithms first and then move on to more advanced algorithms.

Let us analyse the dataset using Random Forest and Naive Bayes.

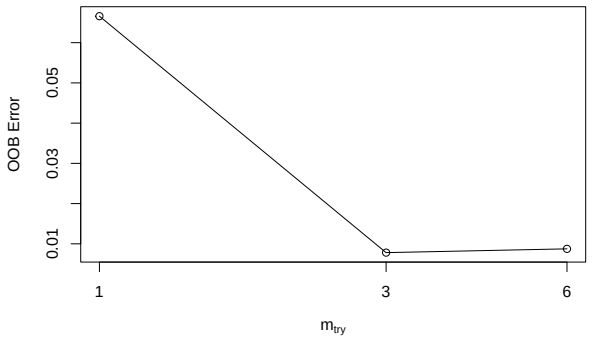
2.2.2 Random Forest

For analysis using Random Forest, we will use the inbuilt function within the "randomForest" library to tune Random Forest to choose the right "m_{try}" values. We will also use the "matrix notation" instead of the "formula notation" as it is a lot easier to program and a lot quicker in terms of performance.

```
## [1] ====================================

## [1] The Random Forest mtry values and error rates are

## mtry = 3  OOB error = 0.78%
## Searching left ...
## mtry = 6     OOB error = 0.88%
## -0.12 1e-05
## Searching right ...
## mtry = 1     OOB error = 6.66%
## -7.52 1e-05
```




```
## [1] ====================================
```

```
## [1]  Details for Random Forest for the HFP Dataset are:
```

```
## [1] Prediction Type : classification
```

```
## [1] Number of Trees (ntree) : 500
```

```
## [1] mtry value : 3
```

```
## [1] ====================================
```

```
## [1] "The Variables in order of decreasing importance in prediction are:"
```

```
## [1] ========================================
```

Table 2: Heart Failure Prediction - Variable Importance (MeanDecreaseGini)

Col Id	Importance	Names
12	546.78908	time
8	216.15513	serum_creatinine
5	152.16972	ejection_fraction
1	106.88246	age
3	97.79567	creatinine_phosphokinase
9	93.12661	serum_sodium
7	81.76683	platelets
6	18.08898	high_blood_pressure
2	16.07963	anaemia
11	14.56157	smoking
4	14.51402	diabetes
10	12.58222	sex

```
## [1] ========================================
```

```
## [1] The Accuracy of the Predictions are:
```

```
## [1] ==========================================

## [1] 0.995

## [1] ==========================================

## [1] The confusion Matrix for the predictions is:

## [1] "=========================================="

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 548   3
##          1   1 248
##
##                Accuracy : 0.995
##                  95% CI : (0.9872, 0.9986)
##     No Information Rate : 0.6862
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9884
##
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.9982
##             Specificity : 0.9880
##          Pos Pred Value : 0.9946
##          Neg Pred Value : 0.9960
##              Prevalence : 0.6863
##          Detection Rate : 0.6850
##    Detection Prevalence : 0.6887
##       Balanced Accuracy : 0.9931
##
##        'Positive' Class : 0
##

## [1] ==========================================
```

We have amazing results using Random Forest, It is not very surprising given how well the dataset is populated and structured.

To understand more about using Random Forest for Classification Problems and how the Variable Importance is calculated, have a look at :-

https://cran.r-project.org/web/packages/randomForest/randomForest.pdf

https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#workings

2.2.3 Naive Bayes

Let us look at Naive Bayes next. We will make some changes to the default parameters based on our observations of the distributions of the variables. We will use the Kernel Distribution Estimation (KDE) for all our continuous variables instead of using the Gaussian Distribution.

```
##
## ================================ Naive Bayes ================================
##
## - Call: naive_bayes.default(x = hfp_cv_train_set[, 1:12], y = hfp_cv_train_set[,      13], laplace =
## - Laplace: 1
## - Classes: 2
## - Samples: 3199
## - Features: 12
## - Conditional distributions:
##     - Bernoulli: 5
##     - KDE: 7
## - Prior probabilities:
##     - 0: 0.6865
##     - 1: 0.3135
##
## -----------------------------------------------------------------------------
```

```
## [1] The Accuracy of the Predictions are:
```

```
## [1] ============================================
```

```
## [1] 0.8825
```

```
## [1] The confusion Matrix for the predictions is:
```

```
## [1] ============================================
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 512  57
##          1  37 194
##
##                Accuracy : 0.8825
##                  95% CI : (0.8581, 0.904)
##     No Information Rate : 0.6862
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7211
##
##  Mcnemar's Test P-Value : 0.05003
##
##             Sensitivity : 0.9326
##             Specificity : 0.7729
##          Pos Pred Value : 0.8998
```

```
##          Neg Pred Value : 0.8398
##              Prevalence : 0.6863
##          Detection Rate : 0.6400
##    Detection Prevalence : 0.7113
##       Balanced Accuracy : 0.8528
##
##        'Positive' Class : 0
##

## [1] ============================================
```

Naive Bayes gives us some decent results with tuning, though the results are not as good as those provided by Random Forest.

For those looking to understand how to tune Naive Bayes using different distributions and estimations. Please have a look at

https://search.r-project.org/CRAN/refmans/naivebayes/html/naive_bayes.html

Here is an extract that provides some detail for immediate reference.

The class "numeric" contains "double" (double precision floating point numbers) and "integer". Depending on the parameters `usekernel` and `usepoisson` different class conditional distributions are applied to columns in the dataset with the class "numeric":

- If `usekernel=FALSE` and `poisson=FALSE` then Gaussian distribution is applied to each "numeric" variable ("numeric"&"integer" or "numeric"&"double")
- If `usekernel=TRUE` and `poisson=FALSE` then kernel density estimation (KDE) is applied to each "numeric" variable ("numeric"&"integer" or "numeric"&"double")
- If `usekernel=FALSE` and `poisson=TRUE` then Gaussian distribution is applied to each "double" vector and Poisson to each "integer" vector. (Gaussian: "numeric" & "double"; Poisson: "numeric" & "integer")
- If `usekernel=TRUE` and `poisson=TRUE` then kernel density estimation (KDE) is applied to each "double" vector and Poisson to each "integer" vector. (KDE: "numeric" & "double"; Poisson: "numeric" & "integer")

By default `usekernel=FALSE` and `poisson=FALSE`, thus Gaussian is applied to each numeric variable.

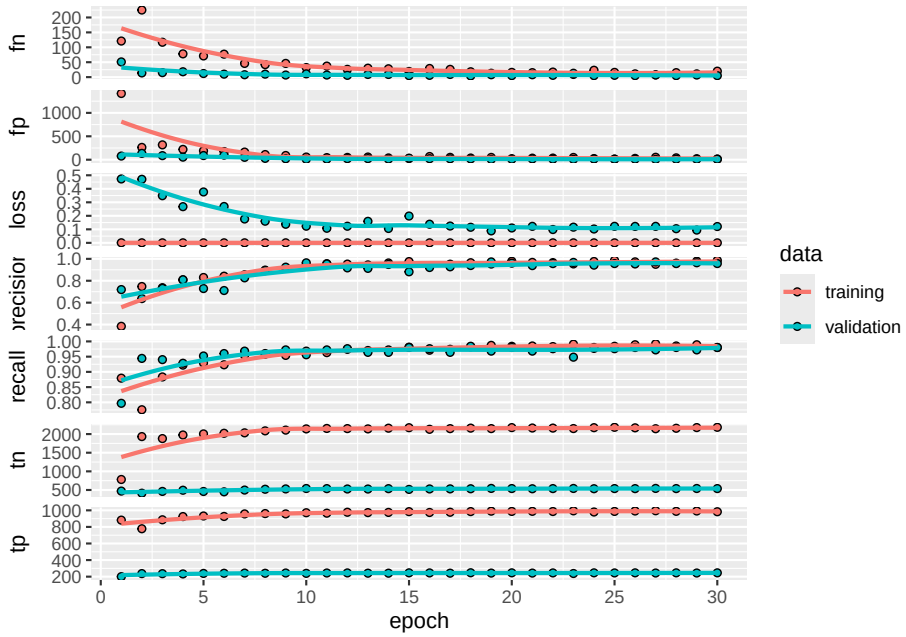
On the other hand, "character", "factor" and "logical" variables are assigned to the Categorical distribution with Bernoulli being its special case.

2.2.4 Artificial Neural Networks (ANN)

Let us investigate using Artificial Neural Networks (ANN) next. For ANN, we will use the Keras 3 package as explained in the "Introduction" chapter. Since our dataset is very simple, it is a very good candidate for using the Keras Sequential API.

```
## [1] The model metrics and trend during training are
```

```
## [1] ============================================
```




```
## [1] ============================================
```

```
## 25/25 - 0s - 2ms/step
```

```
## [1] ============================================
```

```
## [1] The Accuracy of the Predictions is:  0.98
```

```
## [1] ============================================
```

```
## [1] ============================================
```

```
## [1] deaths detected : 246
```

```
## [1] deaths missed : 5
```

```
## [1] live cases flagged : 11
```

```
## [1] ============================================
```

```
## [1] The confusion Matrix for the predictions is:

## [1] ============================================

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 538   5
##          1  11 246
##
##                Accuracy : 0.98
##                  95% CI : (0.9677, 0.9885)
##     No Information Rate : 0.6862
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9539
##
##  Mcnemar's Test P-Value : 0.2113
##
##             Sensitivity : 0.9800
##             Specificity : 0.9801
##          Pos Pred Value : 0.9908
##          Neg Pred Value : 0.9572
##              Prevalence : 0.6863
##          Detection Rate : 0.6725
##    Detection Prevalence : 0.6787
##       Balanced Accuracy : 0.9800
##
##        'Positive' Class : 0
##

## [1] ============================================
```

We have pretty good results with ANN, though they are not as good as what Random Forest provides.

2.3 Predictions for Holdout Set

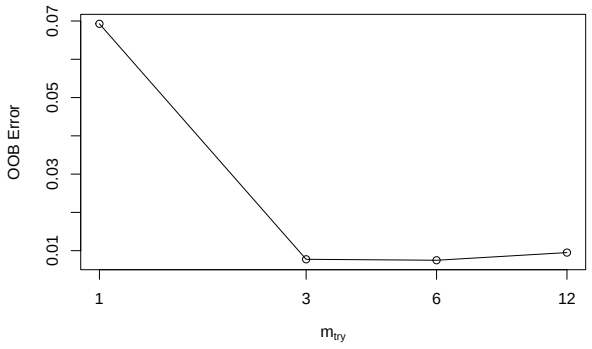
Now let us see how our algorithms do for our Holdout or Test set. We should expect some slight differences in the accuracy of the predictions. We will only evaluate Random Forest and ANN as our choices for the Holdout set.

2.3.1 Random Forest

```
## [1] ====================================
```

```
## [1] The Random Forest mtry values and error rates are
```

```
## mtry = 3  OOB error = 0.78%
## Searching left ...
## mtry = 6     OOB error = 0.75%
## 0.03225806 1e-05
## mtry = 12    OOB error = 0.95%
## -0.2666667 1e-05
## Searching right ...
## mtry = 1     OOB error = 6.93%
## -8.233333 1e-05
```




```
## [1] ====================================
```

```
## [1]  Details for Random Forest  for the HFP Dataset are:
```

```
## [1] Prediction Type:  classification
```

```
## [1] Number of Trees (ntree):  500
```

```
## [1] mtry value:  6
```

```
## [1] ====================================
```

```
## [1] "The Variables in order of decreasing importance in prediction are:"
```

```
## [1] ============================================
```

Table 3: Heart Failure Prediction - Variable Importance (MeanDecreaseGini)

Col Id	Importance	Names
12	781.44896	time
8	281.10882	serum_creatinine
5	173.97751	ejection_fraction
3	109.41887	creatinine_phosphokinase
1	109.24540	age
9	101.08491	serum_sodium
7	88.15502	platelets
6	22.62769	high_blood_pressure
4	15.17608	diabetes
11	14.16734	smoking
2	13.37952	anaemia
10	10.38222	sex

```
## [1] ============================================

## [1] The Accuracy of the predictions is:

## [1] 0.989011

## [1] The confusion Matrix for the predictions is:

## [1] ============================================

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 686  10
##          1   1 304
##
##                Accuracy : 0.989
##                  95% CI : (0.9804, 0.9945)
##     No Information Rate : 0.6863
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9743
##
##  Mcnemar's Test P-Value : 0.01586
##
##             Sensitivity : 0.9985
##             Specificity : 0.9682
##          Pos Pred Value : 0.9856
##          Neg Pred Value : 0.9967
##              Prevalence : 0.6863
##          Detection Rate : 0.6853
```

```
##    Detection Prevalence : 0.6953
##       Balanced Accuracy : 0.9833
##
##        'Positive' Class : 0
##
```

```
## [1] ============================================
```

With Random Forest, we have slightly inferior results compared to the predictions for the validation set but nothing outside of expected values.

2.3.2 Artificial Neural Networks

```
## 32/32 - 0s - 3ms/step
```

```
## [1] ============================================
```

```
## [1] The Accuracy of the Predictions is:
```

```
## [1] 0.982018
```

```
## [1] ============================================
```

```
## [1] ============================================
```

```
## [1] "deaths detected" "301"
```

```
## [1] "deaths missed" "13"
```

```
## [1] "live cases flagged" "5"
```

```
## [1] ============================================
```

```
## [1] The confusion Matrix for the predictions is:
```

```
## [1] ============================================
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 682  13
##          1   5 301
##
##                Accuracy : 0.982
##                  95% CI : (0.9717, 0.9893)
##     No Information Rate : 0.6863
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9579
```

```
##
##  Mcnemar's Test P-Value : 0.09896
##
##             Sensitivity : 0.9927
##             Specificity : 0.9586
##          Pos Pred Value : 0.9813
##          Neg Pred Value : 0.9837
##              Prevalence : 0.6863
##          Detection Rate : 0.6813
##    Detection Prevalence : 0.6943
##       Balanced Accuracy : 0.9757
##
##        'Positive' Class : 0
##
```

```
## [1] ============================================
```

The same is true for ANN too, with ANN offering slightly inferior performance compared to the predictions for the validation set.

2.4 Results & Inference

```
## [1] The Survival Rate of patients in the HFP dataset is :
## [2] 0.6864
```

Table 4: Heart Failure Prediction - Results Summary

Model	overall accuracy	balanced accuracy	deaths detected	deaths missed	live flagged
Random Forest	0.989	0.9833	304	10	1
ANN-SequentialAPI	0.982	0.9757	301	13	5

It has been rather easy so far with a very well structured and very easy to predict dataset. We have also seen that even with a dataset this simple, different Algorithms provide predictions with different accuracies.

Naive Bayes without tuning (using defaults) performs quite poorly but can be improved with tuning.

ANN are pretty good though they are a lot more intensive in terms of programming effort as well as computational needs. We can build ANN which are more accurate but again will require additional efforts.

Random Forest provides us with very good results and is quite simple to program and does not require as much computational resources as ANN. In terms of the returns obtained for the efforts spent, it is the most optimal choice for this dataset.

3 Myocardial Infarction Complications (MIC)

We will pick up the Myocardial Infarction Complications (MIC) or Heart Attack Prediction dataset next.

3.1 Dataset Introduction

A very succinct description of the dataset and it contents is provided in References [3] and [4].

The description can be downloaded from

https://figshare.le.ac.uk/ndownloader/files/22803695

3.1.1 List of Attributes/Features

The list of features is extremely long. Readers who are interested can have a look at Appendix A - MIC Dataset Features & Outcomes for more details about them.

The dataset has 1700 observations and 112 Features.

The very first feature in column 1 is the Record ID (ID).

---

1. Record ID (ID): Unique identifier. Cannot be related to participant. It can be used for reference only.

Type : Numeric/ Integer

It can be disregarded giving us 111 features to work with. Not all observations have all features.

One of the important characteristics of this dataset is that all categorical and the only partially categorical feature can be treated as Ordinals (Ordered Factors) whose numerical value can be used as-is or can be dummy encoded using cumulative encoding.

It also has 11 "complications" for outcomes resulting from MIC. These are Non-Lethal

It also has 1 "complication" for Lethal outcome coded in column 124. The Lethal outcome is further categorised as :-

---

124. Lethal outcome (cause) (LET_IS):

Type: Category (Not Ordered)

Value	Represents	Cases	Fraction
0	unknown (alive)	1429	84.06%
1	cardiogenic shock	110	6.47%
2	pulmonary edema	18	1.06%
3	myocardial rupture	54	3.18%
4	progress of congestive heart failure	23	1.35%
5	thromboembolism	12	0.71%
6	asystole	27	1.59%

7	ventricular fibrillation	27	1.59%
	Missing		0.0%

3.1.2 Problems to solve

In general columns 2-112 can be used as input data for prediction. Possible complications (outputs) are listed in columns 113-124. There are four possible time moments for complication prediction: based on the information known at

1. the time of admission to hospital: all input columns (2-112) except 93, 94, 95, 100, 101, 102, 103, 104, 105 can be used for prediction;

2. the end of the first day (24 hours after admission to the hospital): all input columns (2-112) except 94, 95, 101, 102, 104, 105 can be used for prediction;

3. the end of the second day (48 hours after admission to the hospital) all input columns (2-112) except 95, 102, 105 can be used for prediction;

4. the end of the third day (72 hours after admission to the hospital) all input columns (2-112) can be used for prediction.

3.2 Analysis

3.2.1 Feature Selection

For our Initial Analysis, we will exclude features where there is a lot of missing information. We will exclude the following :-

89. Serum CPK content (KFK_BLOOD) (IU/L) - Which has 99.76% of Missing Values. Imputation makes little sense.

We will still include the Columns below because, imputation using Random Forest seems to work for them.

8. Heredity on CHD (IBS_NASL) - 95.26% of Missing Values.

35. Systolic blood pressure according to Emergency Cardiology Team (S_AD_KBRIG) (mmHg) - 63.29% of Missing Values

36. Diastolic blood pressure according to Emergency Cardiology Team (D_AD_KBRIG) (mmHg) - 63.29% of Missing Values

Extremely Unbalanced Predictors

There are a few predictors which are extremely unbalanced where "Yes" values are lower than 1%of the total and in most cases are also lower than the "Missing" values.

14. Premature atrial contractions in the anamnesis (**nr_01**):

No: Yes: Missing :: 1675: 4: 21

18. Ventricular fibrillation in the anamnesis (**nr_07**):

No: Yes: Missing :: 1678: 1: 21

In it's current form, there is no way to cross check any prediction made based on the "Yes" value.

19. Ventricular paroxysmal tachycardia in the anamnesis (**nr_08**):

No: Yes: Missing :: 1675: 4: 21

In it's current form, there is very little possibility of the "Yes" value appearing on the "Training", "Validation" and "Testing" sets.

20. First-degree AV block in the anamnesis (**np__01**):

No: Yes: Missing :: 1680: 2: 18

In it's current form, there is no possibility of the "Yes" value appearing on the "Training", "Validation" and "Testing" sets.

21. Third-degree AV block in the anamnesis (**np__04**):

No: Yes: Missing :: 1679: 3: 18

In it's current form, there is very little possibility of the "Yes" value appearing on the "Training", "Validation" and "Testing" sets.

22. LBBB (anterior branch) in the anamnesis (**np__05**):

No: Yes: Missing :: 1671: 11: 18

23. Incomplete LBBB in the anamnesis (**np__07**):

No: Yes: Missing :: 1681: 1: 18

In it's current form, there is no way to cross check any prediction made based on the "Yes" value.

24. Complete LBBB in the anamnesis (**np__08**):

No: Yes: Missing :: 1676: 6: 18

25. Incomplete RBBB in the anamnesis (**np__09**):

No: Yes: Missing :: 1680: 2: 18

In it's current form, there is no possibility of the "Yes" value appearing on the "Training", "Validation" and "Testing" sets.

26. Complete RBBB in the anamnesis (**np__10**):

No: Yes: Missing :: 1679: 3: 18

In it's current form, there is very little possibility of the "Yes" value appearing on the "Training", "Validation" and "Testing" sets.

29. Thyrotoxicosis in the anamnesis (**endocr__03**):

No: Yes: Missing :: 1677: 13: 10

33. Chronic pneumonia in the anamnesis (**zab__leg__04**):

No: Yes: Missing :: 1684: 9: 7

34. Pulmonary tuberculosis in the anamnesis (**zab__leg__06**):

No: Yes: Missing :: 1684: 9: 7

42. Paroxysms of supraventricular tachycardia at the time of admission to intensive care unit, (or at a pre-hospital stage) (**SVT__POST**):

No: Yes: Missing :: 1680: 8: 12

43. Paroxysms of ventricular tachycardia at the time of admission to intensive care unit, (or at a pre-hospital stage) (**GT__POST**):

No: Yes: Missing :: 1680: 8: 12

44. Ventricular fibrillation at the time of admission to intensive care unit, (or at a pre-hospital stage) (**FIB_G_POST**):

No: Yes: Missing :: 1673: 15: 12

53. ECG rhythm at the time of admission to hospital: idioventricular (**ritm_ecg_p_06**):

No: Yes: Missing :: 1547: 1: 152

In its current form, there is no way to cross check any prediction made based on the "Yes" value

57. Frequent premature atrial contractions on ECG at the time of admission to hospital (**n_r_ecg_p_02**):

No: Yes: Missing :: 1577: 8: 115

62. Paroxysms of supraventricular tachycardia on ECG at the time of admission to hospital (**n_r_ecg_p_08**):

No: Yes: Missing :: 1581: 4: 115

In its current form, there is very little possibility of the "Yes" value appearing on the "Training", "Validation" and "Testing" sets.

63. Paroxysms of ventricular tachycardia on ECG at the time of admission to hospital (**n_r_ecg_p_09**):

No: Yes: Missing :: 1583: 2: 115

In its current form, there is no possibility of the "Yes" value appearing on the "Training", "Validation" and "Testing" sets.

64. Ventricular fibrillation on ECG at the time of admission to hospital (**n_r_ecg_p_10**):

No: Yes: Missing :: 1583: 2: 115

*Note: In its current form, there is no possibility of the "Yes" value appearing on the "Training", "Validation" and "Testing" sets.

65. Sinoatrial block on ECG at the time of admission to hospital (**n_p_ecg_p_01**):

No: Yes: Missing :: 1583: 2: 115

*Note: In its current form, there is no possibility of the "Yes" value appearing on the "Training", "Validation" and "Testing" sets.

67. Type 1 Second-degree AV block (Mobitz I/ Wenckebach) on ECG at the time of admission to hospital (**n_p_ecg_p_04**):

No: Yes: Missing :: 1580: 5: 115

68. Type 2 Second-degree AV block (Mobitz II/Hay) on ECG at the time of admission to hospital (**n_p_ecg_p_05**):

No: Yes: Missing :: 1583: 2: 115

*Note: In its current form, there is no possibility of the "Yes" value appearing on the "Training", "Validation" and "Testing" sets.

71. LBBB (posterior branch) on ECG at the time of admission to hospital (**n_p_ecg_p_08**):

No: Yes: Missing :: 1578: 7: 115

72. Incomplete LBBB on ECG at the time of admission to hospital (**n_p_ecg_p_09**):

No: Yes: Missing :: 1575: 10: 115

76. Fibrinolytic therapy by Celiasum 750k IU (**fibr_ter_01**):

No: Yes: Missing :: 1677: 13: 10

77. Fibrinolytic therapy by Celiasum 1m IU (**fibr_ter_02**):

No: Yes: Missing :: 1674: 16: 10

79. Fibrinolytic therapy by Streptase (**fibr_ter_05**):

No: Yes: Missing :: 1686: 4: 10

*Note: In its current form, there is very little possibility of the "Yes" value appearing on the "Training", "Validation" and "Testing" sets.

80. Fibrinolytic therapy by Celiasum 500k IU (**fibr_ter_06**):

No: Yes: Missing :: 1681: 9: 10

81. Fibrinolytic therapy by Celiasum 250k IU (**fibr_ter_07**):

No: Yes: Missing :: 1684: 6: 10

82. Fibrinolytic therapy by Streptodecase 1.5m IU (**fibr_ter_08**):

No: Yes: Missing :: 1688: 2: 10

*Note: In its current form, there is no possibility of the "Yes" value appearing on the "Training", "Validation" and "Testing" sets.

We will track these features and zero them out if they affect the predictions negatively.

3.2.2 Outcome Definition

We will only try to predict the Lethal Outcome(LET_IS) and not the other Complications. Otherwise we will need to partition our dataset for each Complication to ensure fair prediction. Predictions for each Complication will need to analysed separately and the predictions cannot be reassembled together into a common dataset easily.

For the Lethal Outcome (LET_IS), we will predict the outcome using 2 methods

1. Binary Method:- A very simple method, where we only predict if the Patient survives or not. Lethal or Death events due to all causes are treated the same.
2. Categorical Method:- Much more complex and error prone. Here we will try to predict the Lethal or Death event and the cause of death viz.
    - 1 - cardiogenic shock
    - 2 - pulmonary edema
    - 3 - myocardial rupture
    - 4 - progress of congestive heart failure
    - 5 - thromboembolism
    - 6 - asystole
    - 7 - ventricular fibrillation

3.2.3 Dataset Preparation

Imputation

For Observations (Rows) where we have missing information, we will need a way to ensure that the Features (Columns) which have no missing information are still included in our predictions. In reality, we would need to have Subject Matter Expertise in CVD before we can impute any value because the Features are all related to each other and imputing values must be done taking these relationships into account.

However for the purposes of this Project, which is more academic in nature and whose results are not very relevant for the choice of treatment or outcome, we will use statistical methods to impute values.

Though not a main topic for this report, Imputation is extremely important as inaccurately imputed values can wreak havoc and lead to inaccurate predictions causing more problems than they solve

We will explore how to the accuracy of our predictions changes with and without Imputation. We will use Naive Bayes for exploratory analysis as it can easily work with missing values when compared to other Algorithms.

Data download and variable set creation

We will download the dataset from the University of Leicester's repository. As indicated in the Introduction section, the dataset is available in CSV format and can easily be imported into a data frame for further processing.

We will create lists of Continuous, Ordinal (Ordered Factors), Partially Ordinal (Unordered Factors) and Nominal (Binary) features so that it is easier to process them later. Binary features are coded as Unordered Factors.

Though using column numbers is easier for cross reference, using column numbers becomes very unwieldy soon enough.

We will also create feature sets or groupings of features that we can use with the Keras Functional API. The feature set groupings are done with very rudimentary knowledge of the features based on their descriptions and the availability of time. In actual practice, we can greatly enhance the accuracy of the predictions with the services of a Subject Matter Expert (SME) in CVD who can help us group features that can be combined together to offer the best possible predictions.

The Reader is encouraged to experiment with the feature sets or work with an SME to build better sets that can offer more accurate predictions.

Let us have a preliminary look at the imported dataset. We will also record the survival rate of patients for the whole dataset which will serve as reference for evaluating the performance of our Algorithms

```
## [1] ================================================


## [1] "The structure of the MIC dataset as imported is"


## [1] ================================================


## 'data.frame':    1700 obs. of  124 variables:
##  $ ID           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ AGE          : int  77 55 52 68 60 64 70 65 60 77 ...
##  $ SEX          : int  1 1 1 0 1 1 1 1 1 0 ...
##  $ INF_ANAM     : int  2 1 0 0 0 0 1 0 0 2 ...
##  $ STENOK_AN    : int  1 0 0 0 0 1 1 1 0 0 ...
##  $ FK_STENOK    : int  1 0 0 0 0 2 2 1 0 0 ...
##  $ IBS_POST     : int  2 0 2 2 2 1 1 2 2 0 ...
##  $ IBS_NASL     : int  NA 0 NA NA NA NA NA NA NA NA ...
##  $ GB           : int  3 0 2 2 3 0 2 2 2 3 ...
##  $ SIM_GIPERT   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ DLIT_AG      : int  7 0 2 3 7 0 7 7 6 6 ...
##  $ ZSN_A        : int  0 0 0 1 0 0 1 0 0 1 ...
##  $ nr_11        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nr_01        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nr_02        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nr_03        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nr_04        : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ nr_07        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nr_08        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ np_01        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ np_04        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ np_05        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ np_07        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ np_08        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ np_09        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ np_10        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ endocr_01    : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ endocr_02    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ endocr_03    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ zab_leg_01   : int  0 0 0 1 0 0 1 0 0 0 ...
##  $ zab_leg_02   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ zab_leg_03   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ zab_leg_04   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ zab_leg_06   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ S_AD_KBRIG   : int  NA NA 150 NA 190 NA 120 NA 200 NA ...
##  $ D_AD_KBRIG   : int  NA NA 100 NA 100 NA 80 NA 120 NA ...
##  $ S_AD_ORIT    : int  180 120 180 120 160 140 120 145 195 200 ...
##  $ D_AD_ORIT    : int  100 90 100 70 90 90 80 95 120 100 ...
##  $ O_L_POST     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ K_SH_POST    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MP_TP_POST   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ SVT_POST     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GT_POST      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ FIB_G_POST   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ant_im       : int  1 4 4 0 4 1 0 0 0 4 ...
##  $ lat_im       : int  0 1 1 1 1 1 0 0 0 1 ...
##  $ inf_im       : int  0 0 0 1 0 0 3 2 3 0 ...
##  $ post_im      : int  0 0 0 0 0 0 0 0 2 0 ...
##  $ IM_PG_P      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ritm_ecg_p_01: int  0 1 1 1 0 0 1 1 1 0 ...
##  $ ritm_ecg_p_02: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ritm_ecg_p_04: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ritm_ecg_p_06: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ritm_ecg_p_07: int  1 0 0 0 1 1 0 0 0 1 ...
##  $ ritm_ecg_p_08: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ n_r_ecg_p_01 : int  0 0 0 0 0 0 0 0 1 0 ...
##  $ n_r_ecg_p_02 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ n_r_ecg_p_03 : int  0 0 1 0 0 0 0 0 0 1 ...
##  $ n_r_ecg_p_04 : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ n_r_ecg_p_05 : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ n_r_ecg_p_06 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ n_r_ecg_p_08 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ n_r_ecg_p_09 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ n_r_ecg_p_10 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ n_p_ecg_p_01 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ n_p_ecg_p_03 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ n_p_ecg_p_04 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ n_p_ecg_p_05 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ n_p_ecg_p_06 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ n_p_ecg_p_07 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ n_p_ecg_p_08 : int  1 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ n_p_ecg_p_09 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ n_p_ecg_p_10 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ n_p_ecg_p_11 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ n_p_ecg_p_12 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ fibr_ter_01  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ fibr_ter_02  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ fibr_ter_03  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ fibr_ter_05  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ fibr_ter_06  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ fibr_ter_07  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ fibr_ter_08  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GIPO_K       : int  0 1 0 1 1 NA NA 0 NA NA ...
##  $ K_BLOOD      : num  4.7 3.5 4 3.9 3.5 NA NA 4.5 NA NA ...
##  $ GIPER_NA     : int  0 0 0 0 0 NA NA 0 NA NA ...
##  $ NA_BLOOD     : int  138 132 132 146 132 NA NA 136 NA NA ...
##  $ ALT_BLOOD    : num  NA 0.38 0.3 0.75 0.45 0.45 0.3 NA 0.3 0.38 ...
##  $ AST_BLOOD    : num  NA 0.18 0.11 0.37 0.22 0.22 0.11 NA 0.37 0.11 ...
##  $ KFK_BLOOD    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ L_BLOOD      : num  8 7.8 10.8 NA 8.3 7.2 11.1 6.2 6.2 6.9 ...
##  $ ROE          : int  16 3 NA NA NA 2 5 20 3 30 ...
##  $ TIME_B_S     : int  4 2 3 2 9 2 1 7 3 3 ...
##  $ R_AB_1_n     : int  0 0 3 0 0 0 0 3 0 0 ...
##  $ R_AB_2_n     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ R_AB_3_n     : int  1 0 0 1 0 0 0 0 0 0 ...
##  $ NA_KB        : int  NA 1 1 NA 0 0 0 0 0 NA ...
##  $ NOT_NA_KB    : int  NA 0 1 NA 0 1 1 0 1 NA ...
##  $ LID_KB       : int  NA 1 1 NA 0 0 0 0 0 NA ...
##  $ NITR_S       : int  0 0 0 0 0 0 0 0 0 0 ...
##   [list output truncated]

## [1] ================================================

## [1] ================================================

## [1] The Survival Rate for patients affected by MIC on this datset is
## [2] 0.8406

## [1] ================================================
```

We will create Training, Testing and CV partitions of the dataset that can be used later by different algorithms with due modifications in the variable types. We will use the dataset with the original values for LET_IS for creating these partitions. We will also set KFK_BLOOD=0 for all observations.

```
## [1] ================================================

## [1] The structure of the MIC dataset after modification of variable types and before imputation is:

## [1] ================================================

## 'data.frame':    1700 obs. of  124 variables:
##  $ ID           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ AGE          : int  77 55 52 68 60 64 70 65 60 77 ...
```

```
##  $ SEX          : Factor w/ 2 levels "0","1": 2 2 2 1 2 2 2 2 2 1 ...
##  $ INF_ANAM     : Ord.factor w/ 4 levels "0"<"1"<"2"<"3": 3 2 1 1 1 1 2 1 1 3 ...
##  $ STENOK_AN    : Ord.factor w/ 7 levels "0"<"1"<"2"<"3"<..: 2 1 1 1 1 2 2 2 1 1 ...
##  $ FK_STENOK    : Ord.factor w/ 5 levels "0"<"1"<"2"<"3"<..: 2 1 1 1 1 3 3 2 1 1 ...
##  $ IBS_POST     : Ord.factor w/ 3 levels "0"<"1"<"2": 3 1 3 3 3 2 2 3 3 1 ...
##  $ IBS_NASL     : Factor w/ 2 levels "0","1": NA 1 NA NA NA NA NA NA NA NA ...
##  $ GB           : Ord.factor w/ 4 levels "0"<"1"<"2"<"3": 4 1 3 3 4 1 3 3 3 4 ...
##  $ SIM_GIPERT   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ DLIT_AG      : Ord.factor w/ 8 levels "0"<"1"<"2"<"3"<..: 8 1 3 4 8 1 8 8 7 7 ...
##  $ ZSN_A        : Factor w/ 5 levels "0","1","2","3",..: 1 1 1 2 1 1 2 1 1 2 ...
##  $ nr_11        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ nr_01        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ nr_02        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ nr_03        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ nr_04        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ nr_07        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ nr_08        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ np_01        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ np_04        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ np_05        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ np_07        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ np_08        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ np_09        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ np_10        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ endocr_01    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ endocr_02    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ endocr_03    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ zab_leg_01   : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 2 1 1 1 ...
##  $ zab_leg_02   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ zab_leg_03   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ zab_leg_04   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ zab_leg_06   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ S_AD_KBRIG   : int  NA NA 150 NA 190 NA 120 NA 200 NA ...
##  $ D_AD_KBRIG   : int  NA NA 100 NA 100 NA 80 NA 120 NA ...
##  $ S_AD_ORIT    : int  180 120 180 120 160 140 120 145 195 200 ...
##  $ D_AD_ORIT    : int  100 90 100 70 90 90 80 95 120 100 ...
##  $ O_L_POST     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ K_SH_POST    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ MP_TP_POST   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ SVT_POST     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ GT_POST      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ FIB_G_POST   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ ant_im       : Ord.factor w/ 5 levels "0"<"1"<"2"<"3"<..: 2 5 5 1 5 2 1 1 1 5 ...
##  $ lat_im       : Ord.factor w/ 5 levels "0"<"1"<"2"<"3"<..: 1 2 2 2 2 2 1 1 1 2 ...
##  $ inf_im       : Ord.factor w/ 5 levels "0"<"1"<"2"<"3"<..: 1 1 1 2 1 1 4 3 4 1 ...
##  $ post_im      : Ord.factor w/ 5 levels "0"<"1"<"2"<"3"<..: 1 1 1 1 1 1 1 1 3 1 ...
##  $ IM_PG_P      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ ritm_ecg_p_01: Factor w/ 2 levels "0","1": 1 2 2 2 1 1 2 2 2 1 ...
##  $ ritm_ecg_p_02: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ ritm_ecg_p_04: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ ritm_ecg_p_06: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ ritm_ecg_p_07: Factor w/ 2 levels "0","1": 2 1 1 1 2 2 1 1 1 2 ...
##  $ ritm_ecg_p_08: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ n_r_ecg_p_01 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 2 1 ...
```

```
##  $ n_r_ecg_p_02 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ n_r_ecg_p_03 : Factor w/ 2 levels "0","1": 1 1 2 1 1 1 1 1 1 2 ...
##  $ n_r_ecg_p_04 : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 1 ...
##  $ n_r_ecg_p_05 : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 1 ...
##  $ n_r_ecg_p_06 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ n_r_ecg_p_08 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ n_r_ecg_p_09 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ n_r_ecg_p_10 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ n_p_ecg_p_01 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ n_p_ecg_p_03 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ n_p_ecg_p_04 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ n_p_ecg_p_05 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ n_p_ecg_p_06 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ n_p_ecg_p_07 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ n_p_ecg_p_08 : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 1 ...
##  $ n_p_ecg_p_09 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ n_p_ecg_p_10 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ n_p_ecg_p_11 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ n_p_ecg_p_12 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ fibr_ter_01  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ fibr_ter_02  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ fibr_ter_03  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ fibr_ter_05  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ fibr_ter_06  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ fibr_ter_07  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ fibr_ter_08  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ GIPO_K       : Factor w/ 2 levels "0","1": 1 2 1 2 2 NA NA 1 NA NA ...
##  $ K_BLOOD      : num  4.7 3.5 4 3.9 3.5 NA NA 4.5 NA NA ...
##  $ GIPER_NA     : Factor w/ 2 levels "0","1": 1 1 1 1 1 NA NA 1 NA NA ...
##  $ NA_BLOOD     : int  138 132 132 146 132 NA NA 136 NA NA ...
##  $ ALT_BLOOD    : num  NA 0.38 0.3 0.75 0.45 0.45 0.3 NA 0.3 0.38 ...
##  $ AST_BLOOD    : num  NA 0.18 0.11 0.37 0.22 0.22 0.11 NA 0.37 0.11 ...
##  $ KFK_BLOOD    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ L_BLOOD      : num  8 7.8 10.8 NA 8.3 7.2 11.1 6.2 6.2 6.9 ...
##  $ ROE          : int  16 3 NA NA NA 2 5 20 3 30 ...
##  $ TIME_B_S     : Ord.factor w/ 9 levels "1"<"2"<"3"<"4"<..: 4 2 3 2 9 2 1 7 3 3 ...
##  $ R_AB_1_n     : Ord.factor w/ 4 levels "0"<"1"<"2"<"3": 1 1 4 1 1 1 1 4 1 1 ...
##  $ R_AB_2_n     : Ord.factor w/ 4 levels "0"<"1"<"2"<"3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ R_AB_3_n     : Ord.factor w/ 4 levels "0"<"1"<"2"<"3": 2 1 1 2 1 1 1 1 1 1 ...
##  $ NA_KB        : Factor w/ 2 levels "0","1": NA 2 2 NA 1 1 1 1 1 NA ...
##  $ NOT_NA_KB    : Factor w/ 2 levels "0","1": NA 1 2 NA 1 2 2 1 2 NA ...
##  $ LID_KB       : Factor w/ 2 levels "0","1": NA 2 2 NA 1 1 1 1 1 NA ...
##  $ NITR_S       : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##   [list output truncated]

## [1] ================================================
```

For imputation of missing values, we will use the Multivariate Imputation by Chained Equations (mice) package. Within mice, we will use Random Forest as the imputation algorithm. The choice of Random Forest is based on experimentation and observation of the resulting imputations.

We will also compare the distributions before and after imputation to be sure that the imputation is not causing large changes in the shape of the distributions.

To inform mice about our variables, we will modify our Nominal(Binary) and Ordinal(Categorical) variables as factors. Continuous variables are already modified as Integers or Numericals during the import process.

We will retain the outcome LET_IS as it is. Conversion to factors will cause us to lose its integer values.

We will also reset KFK_BLOOD=0 for all observations to avoid having to redo it for every algorithm.

For Imputation, we will make an assumption that we will have full access to all the Features/Predictors for all data including the "Testing" or "Holdout" set and that the Features/Predictors for the Training and Holdout sets can take on similar values and they can be combined together if required for imputation. The practical aspect of such a scheme is that when we need to analyse data that has not been seen before, we can combine the unseen or unknown data with already known data to impute more plausible values. This is true in practice and hence the assumption is fair indeed.

As a consequence of this assumption, we will perform the imputation of missing values for the whole MIC dataset at one go.

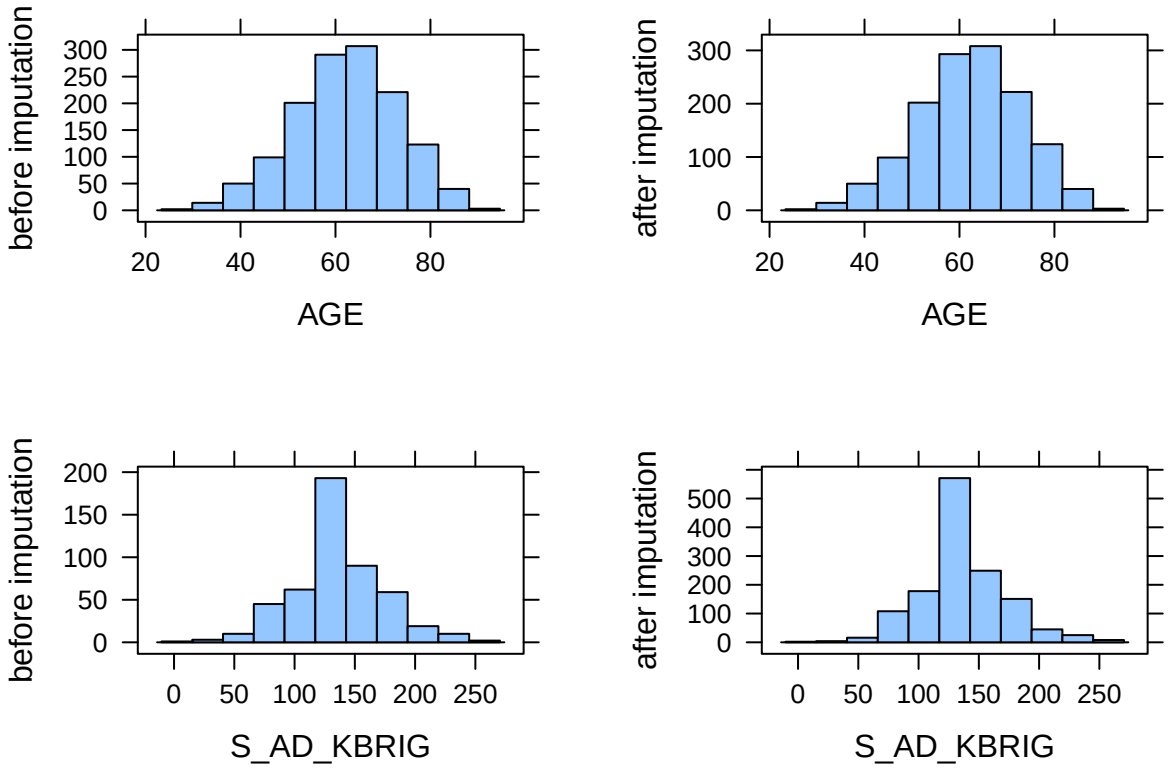
We can partition the dataset into Training, Testing, Training CV and Testing CV sets and perform the imputation individually for each set, However imputation performed for the partitioned datasets only has access to a smaller set of values and imputation fails for several features. This is particularly true for Testing and Testing CV partitions. We can recall that the MIC dataset has only 1700 observations overall for a very complex set of features. Based on our 80:20 partitioning scheme, this leads to 343 observations for the Testing set and 275 observations for the Testing CV set and imputation does not work well for such small numbers of observations.

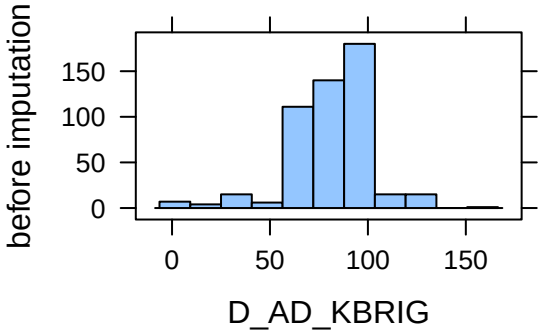
*Note: I have used 75 iterations for the imputation, using 50 causes distortion in the shape of some features, using 95 takes incrementally more time and for some strange reason causes degradation in the accuracy.

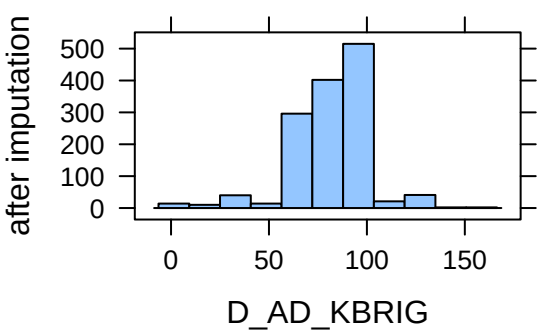
MIC "Training" dataset - distribution of continuous variables before and after imputation

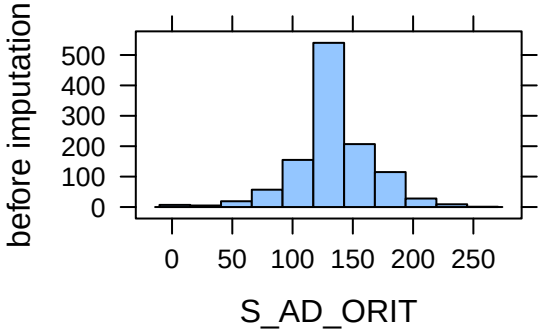
Let us visualise the distributions of our continuous variables before and after imputation.

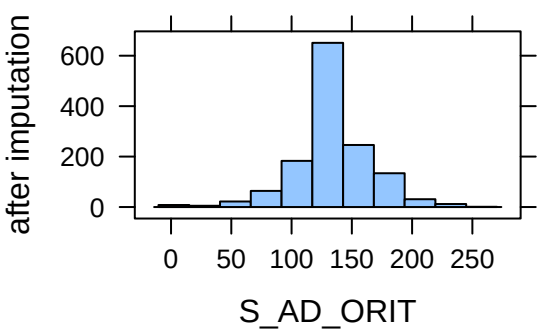
At the end we will also include IBS_NASL which had 95.76% missing values.

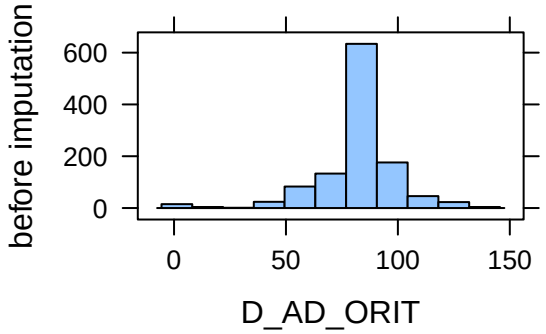


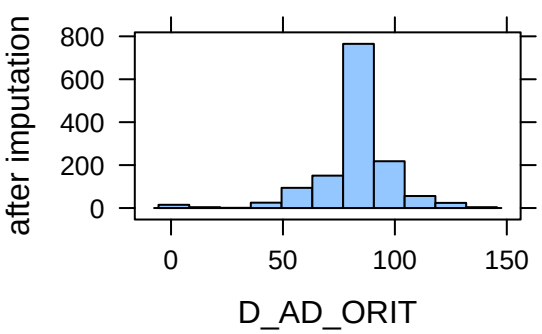

before imputation
150
100
50
0
0
50
100
150
D_AD_KBRIG

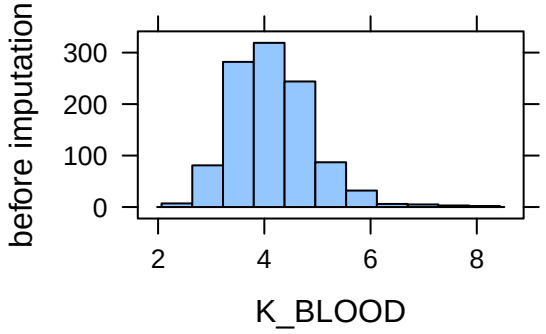

after imputation
500
400
300
200
100
0
0
50
100
150
D_AD_KBRIG

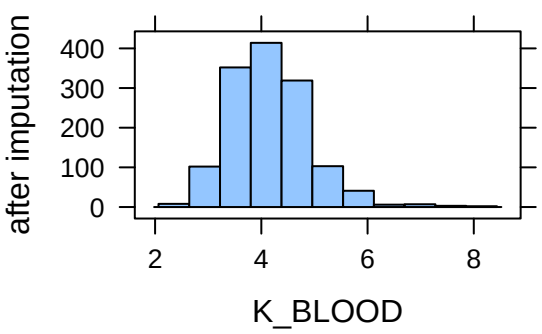

before imputation
500
400
300
200
100
0
0
50
100
150
200
250
S_AD_ORIT

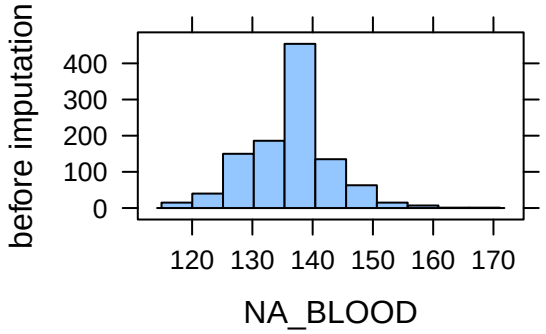

after imputation
600
400
200
0
0
50
100
150
200
250
S_AD_ORIT

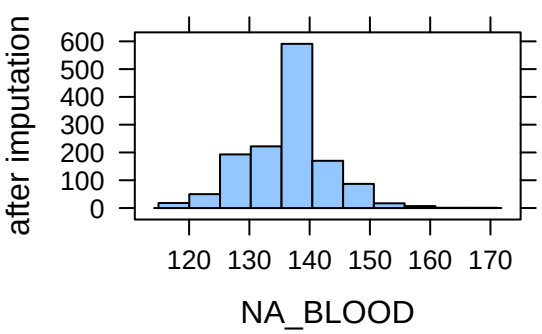

before imputation
600
400
200
0
0
50
100
150
D_AD_ORIT

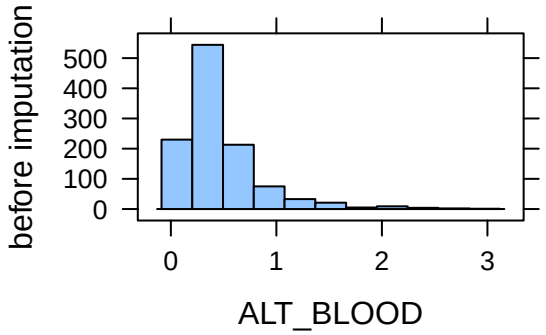

after imputation
800
600
400
200
0
0
50
100
150
D_AD_ORIT

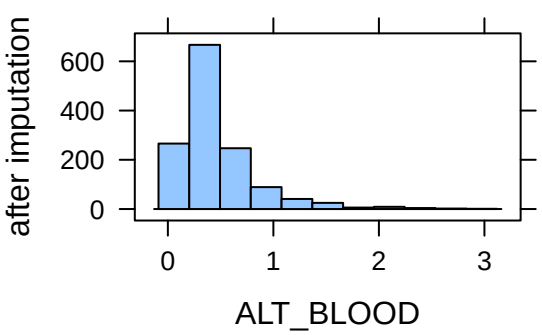

before imputation
300
200
100
0
2
4
6
8
K_BLOOD

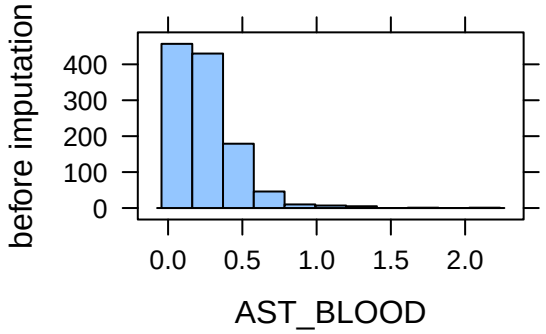

after imputation
400
300
200
100
0
2
4
6
8
K_BLOOD

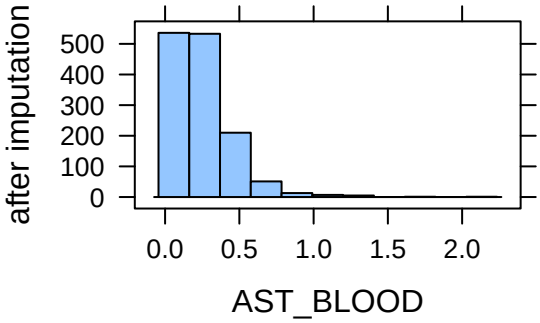

before imputation
400
300
200
100
0
120 130 140 150 160 170
NA_BLOOD

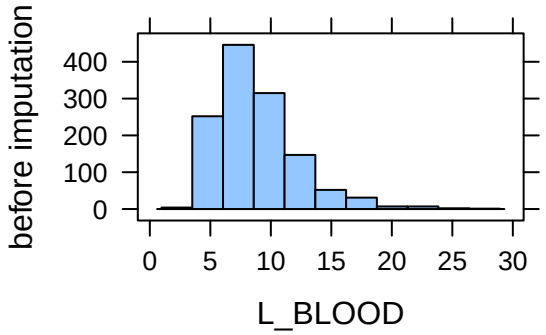

after imputation
600
500
400
300
200
100
0
120 130 140 150 160 170
NA_BLOOD

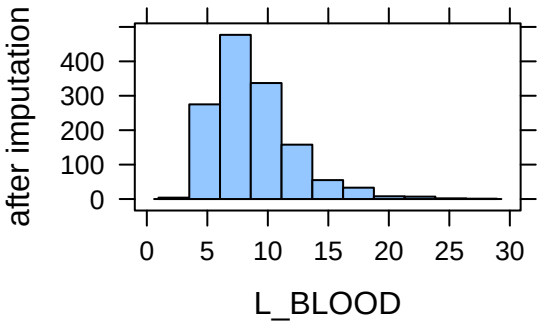

before imputation
500
400
300
200
100
0
0 1 2 3
ALT_BLOOD

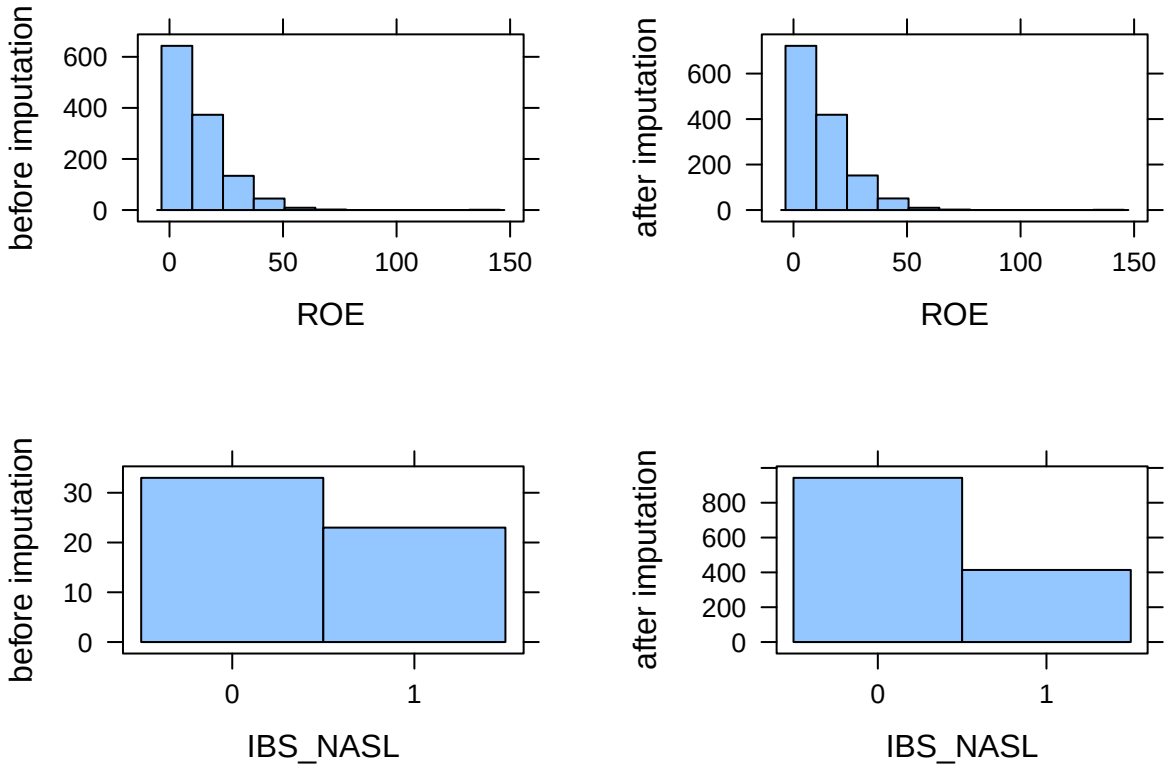

after imputation
600
400
200
0
0 1 2 3
ALT_BLOOD


before imputation
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
AST_BLOOD


after imputation
500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
AST_BLOOD


before imputation
400
300
200
100
0
0
5
10
15
20
25
30
L_BLOOD


after imputation
400
300
200
100
0
0
5
10
15
20
25
30
L_BLOOD



We can see that the MICE package and the RF Algorithm for imputation have done a very good job in preserving the distribution of the continuous variables and the distributions do not look too distorted from the original.

IBS_NASL appears a bit distorted. D_AD_KBRIG has very minor distortion. We will go ahead with both accepting the risk.

3.2.4 Single Category for Deaths

We will try to predict the outcome using the much simpler Binary Method first.

We use Naive Bayes for performing some exploratory analysis of where we can get.

3.2.4.1 Naive Bayes without Imputation - Binary Outcome First up, we will do it without any Imputation. As a reminder, an outcome of 1 represents a lethal outcome and an outcome of 0 represents survival.

For Tuning Naive Bayes, readers can refer back to the Naive Bayes section in the Heart Failure Prediction chapter.

we will configure

- S_AD_KBRIG, S_AD_ORIT, NA_BLOOD, D_AD_KBRIG, D_AD_ORIT and ROE as Poisson
- AGE, K_BLOOD, ALT_BLOOD, AST_BLOOD and L_BLOOD as Gaussian
- Again, KFK_BLOOD = 0 for all observations.

```
##
## ================================== Naive Bayes ==================================
##
## - Call: naive_bayes.default(x = mic_modified_cv_train_set[, 2:112], y = mic_modified_cv_train_set$LE
## - Laplace: 1
## - Classes: 2
## - Samples: 1082
## - Features: 111
## - Conditional distributions:
##     - Bernoulli: 78
##     - Categorical: 21
##     - Poisson: 7
##     - Gaussian: 5
## - Prior probabilities:
##     - 0: 0.8447
##     - 1: 0.1553
##
## --------------------------------------------------------------------------------
```

```
## [1] ====================================
```

```
## [1] The table of predictions is:
```

```
##
##   0   1
## 223  52
```

```
## [1] ====================================
```

```
## [1] ====================================
```

```
## [1] The table of actual values is:
```

```
##
##   0   1
## 229  46
```

```
## [1] ====================================
```

```
## [1] The accuracy of predictions is:
```

```
## [1] 0.8472727
```

```
## [1] ====================================
```

```
## [1] "The confusion matrix is: "
```

```
## [1] ====================================
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 205  18
##          1  24  28
##
##                Accuracy : 0.8473
##                  95% CI : (0.7992, 0.8877)
##     No Information Rate : 0.8327
##     P-Value [Acc > NIR] : 0.2901
##
##                   Kappa : 0.4789
##
##  Mcnemar's Test P-Value : 0.4404
##
##             Sensitivity : 0.8952
##             Specificity : 0.6087
##          Pos Pred Value : 0.9193
##          Neg Pred Value : 0.5385
##              Prevalence : 0.8327
##          Detection Rate : 0.7455
##    Detection Prevalence : 0.8109
##       Balanced Accuracy : 0.7519
##
##        'Positive' Class : 0
##
```

```
## [1] ====================================
```

We are able to detect 28 deaths, while missing 18 and flagging 24 live cases as deaths.

The Balanced Accuracy with Naive Bayes is decent. The Sensitivity is good. Specificity is quite poor comparatively.

Though our Overall Accuracy looks like it is very good, we can see that it is not very far from the "no information rate". We can also recall that the survival rate is around 0.84 for the whole dataset.

3.2.4.2 Naive Bayes with Imputation - Binary Outcome

We will redo our analysis with imputation of missing values.

We will configure

- S_AD_KBRIG, S_AD_ORIT, NA_BLOOD, D_AD_KBRIG, D_AD_ORIT and ROE as Poisson
- AGE, K_BLOOD, ALT_BLOOD, AST_BLOOD and L_BLOOD as Gaussian
- As usual, KFK_BLOOD = 0 for all observations

Let us now check what Naive Bayes with imputation can do.

```
##
## ================================== Naive Bayes ==================================
##
## - Call: naive_bayes.default(x = mic_modified_cv_train_set[, c(2:112)],      y = mic_modified_cv_trai
## - Laplace: 1
## - Classes: 2
## - Samples: 1082
## - Features: 111
## - Conditional distributions:
##     - Bernoulli: 78
##     - Categorical: 21
##     - Poisson: 7
##     - Gaussian: 5
## - Prior probabilities:
##     - 0: 0.8447
##     - 1: 0.1553
##
## --------------------------------------------------------------------------------

## [1] ====================================

## [1] The table of predictions is

##
##   0   1
## 220  55

## [1] ====================================

## [1] ====================================

## [1] The table of actual values is

##
##   0   1
## 229  46

## [1] ====================================
```

```
## [1] The accuracy of predictions is

## [1] 0.8363636

## [1] ====================================

## [1] "The confusion matrix is"

## [1] ====================================

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 202  18
##          1  27  28
##
##                Accuracy : 0.8364
##                  95% CI : (0.7872, 0.8781)
##     No Information Rate : 0.8327
##     P-Value [Acc > NIR] : 0.4749
##
##                   Kappa : 0.4552
##
##  Mcnemar's Test P-Value : 0.2330
##
##             Sensitivity : 0.8821
##             Specificity : 0.6087
##          Pos Pred Value : 0.9182
##          Neg Pred Value : 0.5091
##              Prevalence : 0.8327
##          Detection Rate : 0.7345
##    Detection Prevalence : 0.8000
##       Balanced Accuracy : 0.7454
##
##        'Positive' Class : 0
##

## [1] ====================================
```

Imputation seems to have a slightly negative effect on both the Overall Accuracy and the Balanced Accuracy. There is a small reduction in the Sensitivity.

We are now able to detect 28 deaths while missing 18 and flagging 27 lives cases as deaths.

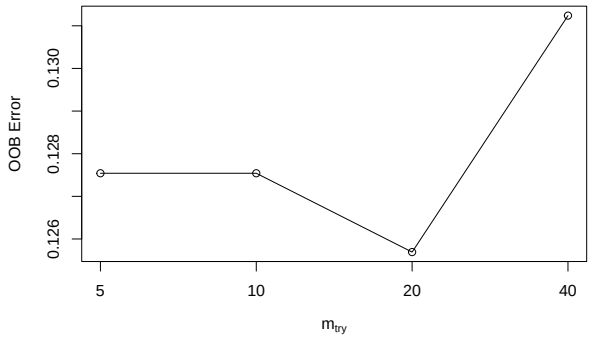
3.2.4.3 Random Forest with Imputation - Binary Outcome Random Forest cannot easily manage missing values and generates errors. Imputation is necessary for us to run the Random Forest algorithm on the dataset. Random Forest is also very sensitive to "noisy" or "irrelevant" data and can generate erroneous predictions which are far away from the target.

Similar to what we did for the HFP dataset, we will use the inbuilt function within the "randomForest" library to tune Random Forest to choose the right "m_{try}" values. We will also use the "matrix notation" instead of the "formula notation" as it is easier to program and quicker in terms of performance.

```
## [1] ====================================

## [1] The Random Forest mtry values and error rates are

## mtry = 10  OOB error = 12.75%
## Searching left ...
## mtry = 20    OOB error = 12.57%
## 0.01449275 1e-06
## mtry = 40    OOB error = 13.12%
## -0.04411765 1e-06
## Searching right ...
## mtry = 5     OOB error = 12.75%
## -0.01470588 1e-06
```




```
## [1] ====================================

## [1] The details for Random Forest are

## [1] Prediction Type : classification

## [1] Number of Trees (ntree) : 500

## [1] mtry value : 20

## [1] ====================================

## [1] ====================================
```

```
## [1] "The table of predictions is"

##
##   0   1
## 263  12

## [1] ====================================

## [1] ====================================

## [1] "The table of actual values is"

##
##   0   1
## 229  46

## [1] ====================================

## [1] "The accuracy of predictions is"

## [1] 0.8690909

## [1] ====================================

## [1] "The confusion matrix is"

## [1] ====================================

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 228  35
##          1   1  11
##
##                Accuracy : 0.8691
##                  95% CI : (0.8234, 0.9066)
##     No Information Rate : 0.8327
##     P-Value [Acc > NIR] : 0.05907
##
##                   Kappa : 0.3332
##
##  Mcnemar's Test P-Value : 3.798e-08
##
##             Sensitivity : 0.9956
##             Specificity : 0.2391
##          Pos Pred Value : 0.8669
##          Neg Pred Value : 0.9167
##              Prevalence : 0.8327
##          Detection Rate : 0.8291
##    Detection Prevalence : 0.9564
##       Balanced Accuracy : 0.6174
##
##        'Positive' Class : 0
##
```

```
## [1] ====================================
```

```
## [1] The first 20 Features in order of decreasing importance in prediction are:
```

```
## [1] ============================================
```

Table 6: MIC Prediction - Variable Importance (MeanDecreaseGini)

Col Id	Importance	Names
37	16.702750	D_AD_ORIT
36	16.225828	S_AD_ORIT
39	15.738331	K_SH_POST
1	13.002171	AGE
89	11.731431	L_BLOOD
90	11.164293	ROE
35	10.406848	D_AD_KBRIG
91	9.553961	TIME_B_S
83	8.887651	K_BLOOD
87	8.685580	AST_BLOOD
85	8.489213	NA_BLOOD
11	8.025292	ZSN_A
34	7.843407	S_AD_KBRIG
99	7.517321	NA_R_1_n
4	6.780125	STENOK_AN
86	6.553242	ALT_BLOOD
44	5.822690	ant_im
45	5.456228	lat_im
92	5.153608	R_AB_1_n
10	4.909142	DLIT_AG

```
## [1] ============================================
```

Though our overall accuracy with Random Forest looks good, The balanced accuracy is quite poor and the number of deaths detected accurately is quite low.

The Sensitivity is extremely good and the Specificity is just as extremely poor.

We can also see the 20 most important Features that Random Forest has used in making its predictions. These provide some good insight into the relative importance of each Feature/Predictor as evaluated by Random Forest.

We can detect 11 deaths while missing 35 and flagging 1 live case inaccurately as a death.

Random Forest seems to generate poor predictions for death events (LET_IS = 1). It is not very useful in the context of our MIC dataset and we will not explore it further.

3.2.4.4 XGBoost

We will look at XGBoost next for Binary Outcome Prediction.

For XGBoost, we will need to use the original MIC data without modifying categorical and binary variables as factors.

As stated earlier in the Additional Notes and Cautions section of the Introduction chapter, Though support for "categoricals" is available in R through factors from XGBoost version 3.0 onwards, the feature is considered experimental. According to the developers, only the Python package is fully supported.

https://xgboost.readthedocs.io/en/stable/tutorials/categorical.html

In this project, I have only used Integer and Numerical values as inputs to XGBoost.

Though this seems incorrect at first glance, a detailed reading of the descriptions of the MIC dataset makes it clear that most of the "Ordinal attribute can be interpreted as numerical. Possible encoding as is or cumulative dummy coding". The only partially ordered attribute is "ZSN_A". Even this is not completely unordered.

XGBoost provides a huge number of tuning parameters. You can read more about the tuning parameters for XGBoost using the help function available within R or at

https://xgboost.readthedocs.io/en/stable/parameter.html

An extract is provided for immediate reference about the parameters used in our model.

- `verbose` [default=1]
  - Verbosity of printing messages. Valid values are 0 (silent), 1 (warning), 2 (info), 3 (debug).If 0, xgboost will stay silent. If 1, it will print information about performance. If 2, some additional information will be printed out.
- `nthread` [default to maximum number of threads available if not set]
  - Number of parallel threads used to run XGBoost. When choosing it, please keep thread contention and hyperthreading in mind
- `nrounds`
  - max number of boosting iterations.
- `eta` [alias: `learning_rate`]
  - Step size shrinkage used in update to prevent overfitting. After each boosting step, we can directly get the weights of new features, and eta shrinks the feature weights to make the boosting process more conservative.
  - range: [0,1]
  - default value: 0.3 for tree-based boosters, 0.5 for linear booster.
- `max_depth` [default=6, type=int32]
  - Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. 0 indicates no limit on depth. Beware that XGBoost aggressively consumes memory when training a deep tree. `exact` tree method requires non-zero value.
  - range: [0,∞]
- `objective`

  - `binary:logistic`: logistic regression for binary classification, output probability

- `eval_metric` [default according to objective]
  - `error`: Binary classification error rate. It is calculated as `#(wrong cases)/#(all cases)`. For the predictions, the evaluation will regard the instances with prediction value larger than 0.5 as positive instances, and the others as negative instances.

- `seed`
  - Random number seed. If not specified, will take a random seed through R's own RNG engine.

For tuning XGBoost, we will set max_depth = 10, eta = 0.15 and nrounds = 200. I have set nthread = 8, the readers and evaluators can change the number of threads to what is convenient to them.

```
## [1] ====================================

## [1] "The details for XGBoost are"

## ##### xgb.Booster
## call:
##   xgb.train(data = dtrain, nrounds = 200, objective = "binary:logistic",
##     verbose = 0, seed = 1024, booster = "gbtree", max_depth = 10,
##     eta = 0.15, nthread = 8)
## # of features: 111
## # of rounds:  200

## [1] ====================================

## [1] ====================================

## [1] The table of predictions is:

##
##   0   1
## 249  26

## [1] ====================================

## [1] ====================================

## [1] The table of actual values is:

##
##   0   1
## 229  46

## [1] ====================================
```

```
## [1] The accuracy of predictions is

## [1] 0.9272727

## [1] The confusion matrix is

## [1] ====================================

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 229  20
##          1   0  26
##
##                Accuracy : 0.9273
##                  95% CI : (0.8899, 0.955)
##     No Information Rate : 0.8327
##     P-Value [Acc > NIR] : 3.128e-06
##
##                   Kappa : 0.6841
##
##  Mcnemar's Test P-Value : 2.152e-05
##
##             Sensitivity : 1.0000
##             Specificity : 0.5652
##          Pos Pred Value : 0.9197
##          Neg Pred Value : 1.0000
##              Prevalence : 0.8327
##          Detection Rate : 0.8327
##    Detection Prevalence : 0.9055
##       Balanced Accuracy : 0.7826
##
##        'Positive' Class : 0
##

## [1] ====================================
```

We can detect 26 deaths while missing 20 and zero live cases flagged as a deaths.

XGBoost provides very good overall accuracy and balanced accuracy which are better that what Naive Bayes provides. We have perfect sensitivity but specificity is quite poor.

The number of deaths that XGBoost is able to predict accurately is also quite good. Where XGBoost really shines bright is in its the ability to avoid prediction of death events as live events. This helps increase both overall accuracy and balanced accuracy.

***Note: The Accuracy of XGBoost reduces quite drastically when one-hot encoding of categorical variables is used for some reason.**

XGBoost is also very robust to the presence of NA values in the observations and can ignore them while using the other predictors. It is also extremely quick. Needless to say, it an excellent choice when quick results are desired.

3.2.4.5 Artificial Neural Networks Let us see if ANN can make our predictions any better than Naive Bayes or XGBoost. Since ANN do not work very well with missing values, we will need to reduce the number of missing values by

1. Imputation
2. Making some educated choices for all observations for a particular feature.
3. Removing observations that cannot be filled in using (1) and (2) as above.

We start exploring ANN by encoding the categorical variables using "Dummy" variables. We will use the dummyVars function from the caret Package as it can support ordinals (ordered factors).

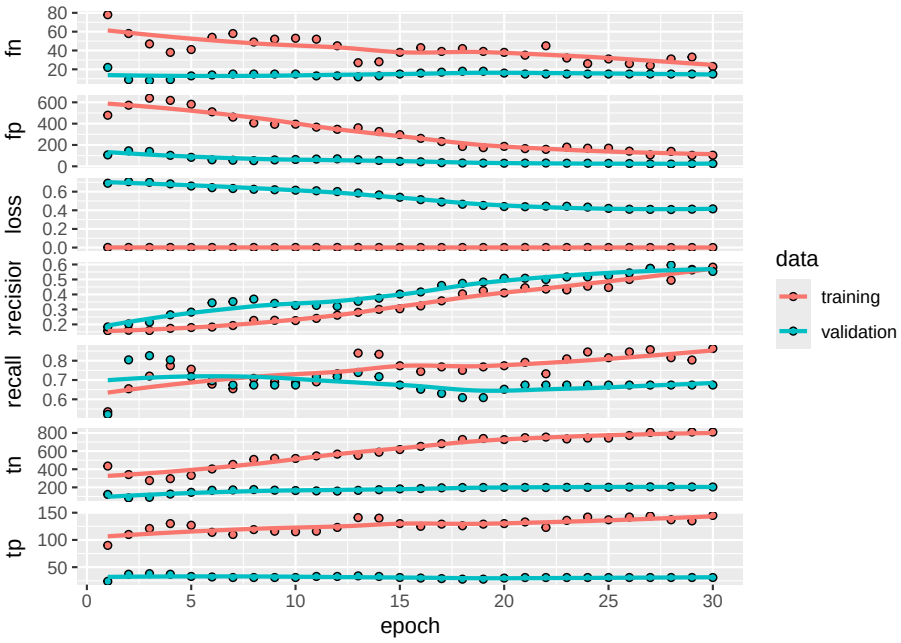
As earlier, we first explore only binary outcomes for LET_IS. 0 indicates survival and 1 indicates a death or lethal event.

3.2.4.5.1 Sequential API

Let us try ANN first with Sequential API

```
## [1] Sequential API - Metrics and Trends during Training
```

```
## [1] ============================================
```




```
## [1] ============================================
```

```
## 9/9 - 0s - 8ms/step
```

```
## [1] ================================
```

```
## [1] The Table of Predicted values is
```

```
##
##   0   1
## 235  40
```

```
## [1] ================================
```

```
## [1] ====================================
```

```
## [1] "The table of actual values is"
```

```
##
##   0   1
## 229  46
```

```
## [1] ====================================
```

```
## [1] ================================
```

```
## [1] "Validation accuracy is : " "0.8545"
```

```
## [1] ================================
```

```
## [1] "deaths detected" "31"
```

```
## [1] "deaths missed" "15"
```

```
## [1] "survival cases flagged as deaths" "25"
```

```
## [1] ================================
```

```
## [1] "The Confusion Matrix is"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 204  15
##          1  25  31
##
##                Accuracy : 0.8545
##                  95% CI : (0.8072, 0.894)
##     No Information Rate : 0.8327
##     P-Value [Acc > NIR] : 0.1881
##
##                   Kappa : 0.5196
##
##  Mcnemar's Test P-Value : 0.1547
##
##             Sensitivity : 0.8908
##             Specificity : 0.6739
##          Pos Pred Value : 0.9315
##          Neg Pred Value : 0.5536
##              Prevalence : 0.8327
##          Detection Rate : 0.7418
##    Detection Prevalence : 0.7964
##       Balanced Accuracy : 0.7824
##
##        'Positive' Class : 0
##
```

```
## [1] ================================
```

We can now detect 31 deaths while missing 15 and flagging 25 live cases as deaths.

With ANN built using Keras sequential API, we are able to better our predictions for deaths. Though the number of live events flagged as deaths is higher than XGBoost or Naive Bayes, our balanced accuracy has also improved. We are seeing some incremental improvements in the Specificity.

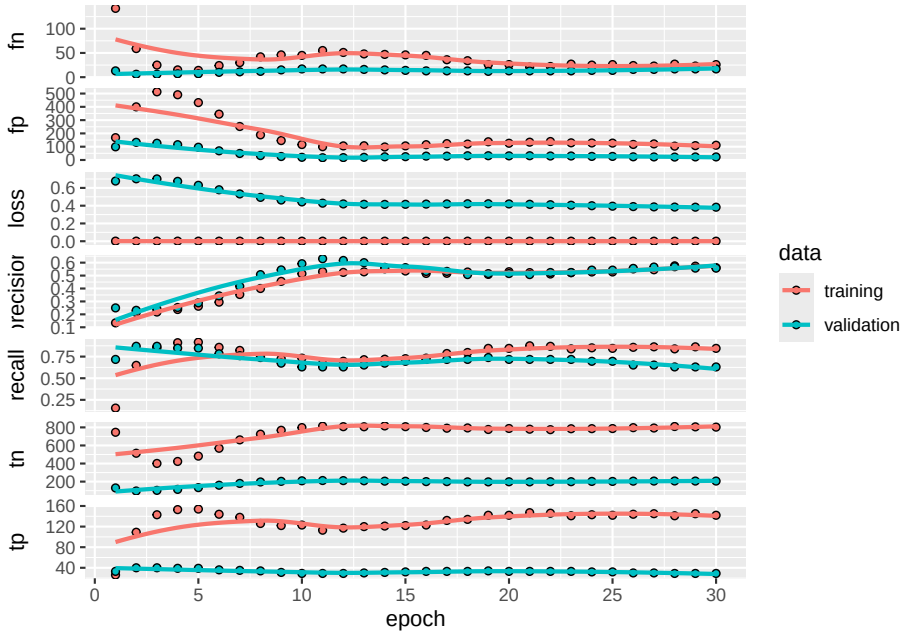
We can further tune the ANN to achieve much higher overall and balanced accuracies, but the flip side is that the ANN perform very poorly when run against other validation or test sets. Just like over-fitting or over-training, we suffer from over-optimising the ANN to the validation set.

The dataset is also very complex and finding holdout and validation subsets that are truly reflective of the whole dataset is a real challenge. Creation of the holdout and validation sets with different seeds and running the ANN against them causes variations in the predictions. So we need to judiciously choose ANN whose predictions do not vary wildly between different sets.

3.2.4.5.2 Functional API We will try ANN with Functional API next. Functional API are extremely versatile and flexible when compared to Sequential API. Functional API allow us to build hierarchies of layers and group features together to provide very good predictions. As indicated earlier, such grouping would require the services of an SME in CVD who can advise on the features that work best when grouped together and can also advice on the hierarchy to be built to get better predictions. We will build something that is quite basic to demonstrate the utility of Functional API.

```
## [1] Functional API - Metrics and Trends during Training
```

```
## [1] ============================================
```




```
## [1] ============================================
```

```
## 9/9 - 0s - 15ms/step
```

```
## [1] ================================
```

```
## [1] The Table of Predicted values is:
```

```
##
##   0   1
## 235  40
```

```
## [1] ================================
```

```
## [1] ====================================
```

```
## [1] The table of actual values is:
```

```
##
##   0   1
## 229  46
```

```
## [1] ====================================
```

```
## [1] ================================
```

```
## [1] Validation accuracy is:  0.8545
```

```
## [1] ================================
```

```
## [1] "deaths detected" "29"
```

```
## [1] "deaths missed" "17"
```

```
## [1] "survival cases flagged as deaths" "23"
```

```
## [1] ================================
```

```
## [1] The Confusion Matrix is
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 206  17
##          1  23  29
##
##                Accuracy : 0.8545
##                  95% CI : (0.8072, 0.894)
##     No Information Rate : 0.8327
##     P-Value [Acc > NIR] : 0.1881
##
##                   Kappa : 0.5037
##
##  Mcnemar's Test P-Value : 0.4292
##
##             Sensitivity : 0.8996
##             Specificity : 0.6304
##          Pos Pred Value : 0.9238
##          Neg Pred Value : 0.5577
##              Prevalence : 0.8327
##          Detection Rate : 0.7491
##    Detection Prevalence : 0.8109
##       Balanced Accuracy : 0.7650
##
##        'Positive' Class : 0
##
```

```
## [1] ================================
```

We are now able to detect 29 deaths while missing 17 and flagging 23 live cases as deaths.

With Keras Functional API, The number of deaths detected has reduced. However, we are able to match results in terms of the overall accuracy as well as reducing the live events marked as deaths. Though not as good as with the Sequential API, we are seeing some incremental improvements in the Specificity.

However, I must reiterate that Functional API work best when the features/predictors are put together to provide the best possible predictions and I must be honest in admitting that this project is the first time that I have tried my hand at ANN and my knowledge of ANN is very basic at best. It is very much possible to build ANN with Functional API that will provide higher overall and balanced accuracies while increasing the number of deaths detected.

3.2.5 Multiple Categories for Deaths

We will now pick up our biggest challenge as yet which is to predict the lethal event (LET_IS) along with the category. With all our algorithms, we are really struggling even with the much simpler binary outcome for predicting the lethal event so our chances of being able to achieve decent categorical accuracy and predictions for the individual categories in the lethal event are likely to be bleak. We will only use XGBoost and ANN to evaluate the same. Readers are encouraged to check with the other algorithms as they deem fit.

For multiple categories for deaths, our goals will also be slightly different. While it is really important to predict outcomes at an individual level accurately, here we will strive to:-

1. Predict outcomes at the level of the category - Though this seems a bit misleading, we must remember that such predictions are still useful for people who are looking at summary statistics at a higher level of abstraction. This should be quite possible.
    1. We will use the term "Shape" (of predictions) to mean the overall number of predictions for each class at the level of the class. This will give us a summary view of the predictions at the class level.
2. Predict individual outcomes within each category - This will require predictions within each category to be accurate, this is going to be very difficult and we are most likely to fail on this aspect, we will see where we can get to.

3.2.5.1 XGBoost

Let us start with XGBoost.

```
## [1] ====================================

## [1] The details for XGBoost are:

## ##### xgb.Booster
## call:
##   xgb.train(params = list(num_class = 8, booster = "gbtree"), data = dtrain_categorical,
##     nrounds = 200, objective = "multi:softmax", verbose = 0,
##     max_depth = 10, eta = 0.15, nthread = 8)
## # of features: 111
## # of rounds:  200

## [1] "===================================="

## [1] ====================================

## [1] The table of predictions is:
```

```
##
##   0   1   2   3   6
## 256  14   1   3   1
```

```
## [1] ====================================
```

```
## [1] ====================================
```

```
## [1] The table of actual values is:
```

```
##
##   0   1   2   3   4   5   6   7
## 229  18   3   9   4   2   5   5
```

```
## [1] ====================================
```

```
## [1] The accuracy of predictions is:
```

```
## [1] 0.8727273
```

```
## [1] ====================================
```

```
## [1] "The confusion matrix is"
```

```
## [1] ====================================
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1   2   3   4   5   6   7
##          0 229   6   2   7   4   2   2   4
##          1   0  10   1   1   0   0   1   1
##          2   0   0   0   0   0   0   1   0
##          3   0   1   0   1   0   0   1   0
##          4   0   0   0   0   0   0   0   0
##          5   0   0   0   0   0   0   0   0
##          6   0   1   0   0   0   0   0   0
##          7   0   0   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.8727
##                  95% CI : (0.8275, 0.9097)
##     No Information Rate : 0.8327
##     P-Value [Acc > NIR] : 0.04136
##
##                   Kappa : 0.4241
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            1.0000  0.55556 0.000000 0.111111  0.00000 0.000000
## Specificity            0.4130  0.98444 0.996324 0.992481  1.00000 1.000000
## Pos Pred Value         0.8945  0.71429 0.000000 0.333333      NaN      NaN
## Neg Pred Value         1.0000  0.96935 0.989051 0.970588  0.98545 0.992727
## Prevalence             0.8327  0.06545 0.010909 0.032727  0.01455 0.007273
## Detection Rate         0.8327  0.03636 0.000000 0.003636  0.00000 0.000000
## Detection Prevalence   0.9309  0.05091 0.003636 0.010909  0.00000 0.000000
## Balanced Accuracy      0.7065  0.77000 0.498162 0.551796  0.50000 0.500000
##                      Class: 6 Class: 7
## Sensitivity          0.000000  0.00000
## Specificity          0.996296  1.00000
## Pos Pred Value       0.000000      NaN
## Neg Pred Value       0.981752  0.98182
## Prevalence           0.018182  0.01818
## Detection Rate       0.000000  0.00000
## Detection Prevalence 0.003636  0.00000
## Balanced Accuracy    0.498148  0.50000

## [1] ====================================

## [1] ================================

## [1] The Number of Deaths correctly categorised is :

## [1] 11

## [1] ================================

## [1] ================================

## [1] The Number of Deaths correctly detected is :

## [1] 19

## [1] ================================
```

With XGBoost, we are able to predict and classify 11 deaths accurately. We are able to detect 19 deaths while missing 27 deaths.

As expected, XGBoost is not as accurate with predictions for multiple categories of outcomes as it was for the binary outcome.

As usual, it is extremely quick.

3.2.5.2 Artificial Neural Networks

3.2.5.2.1 Sequential API Let us see what ANN built using the Sequential API can do

```
## 9/9 - 0s - 23ms/step

## [1] ================================

## [1] The Table of Predicted values is

##
##   0   1   2   3   4   5   6   7
## 221  25   2  13   5   2   4   3

## [1] ====================================

## [1] The table of actual values is:

##
##   0   1   2   3   4   5   6   7
## 229  18   3   9   4   2   5   5

## [1] ================================

## [1] Validation accuracy is :

## [1] 0.7781818

## [1] ================================

## [1] ================================

## [1] The Confusion Matrix is

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1   2   3   4   5   6   7
##          0 202   4   0   7   1   1   3   3
##          1   8  10   2   2   1   0   1   1
##          2   1   0   1   0   0   0   0   0
##          3   7   3   0   0   1   1   0   1
##          4   3   0   0   0   1   0   1   0
##          5   2   0   0   0   0   0   0   0
##          6   3   1   0   0   0   0   0   0
##          7   3   0   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.7782
```

```
##                  95% CI : (0.7244, 0.8259)
##     No Information Rate : 0.8327
##     P-Value [Acc > NIR] : 0.9923
##
##                   Kappa : 0.312
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.8821  0.55556 0.333333  0.00000 0.250000 0.000000
## Specificity            0.5870  0.94163 0.996324  0.95113 0.985240 0.992674
## Pos Pred Value         0.9140  0.40000 0.500000  0.00000 0.200000 0.000000
## Neg Pred Value         0.5000  0.96800 0.992674  0.96565 0.988889 0.992674
## Prevalence             0.8327  0.06545 0.010909  0.03273 0.014545 0.007273
## Detection Rate         0.7345  0.03636 0.003636  0.00000 0.003636 0.000000
## Detection Prevalence   0.8036  0.09091 0.007273  0.04727 0.018182 0.007273
## Balanced Accuracy      0.7345  0.74859 0.664828  0.47556 0.617620 0.496337
##                      Class: 6 Class: 7
## Sensitivity           0.00000  0.00000
## Specificity           0.98519  0.98889
## Pos Pred Value        0.00000  0.00000
## Neg Pred Value        0.98155  0.98162
## Prevalence            0.01818  0.01818
## Detection Rate        0.00000  0.00000
## Detection Prevalence  0.01455  0.01091
## Balanced Accuracy     0.49259  0.49444

## [1] ================================

## [1] ================================

## [1] The Number of Deaths correctly categorised is :

## [1] 12

## [1] ================================

## [1] ================================

## [1] The Number of Deaths correctly detected is :

## [1] 27

## [1] ================================
```

When we need to predict categorical outcomes with ANN, the need for modifying the class weights becomes apparent. It is a bit of a paradoxical situation because by modifying the class weights we can get almost any ANN to predict the shape or categorical distributions a lot more accurately than individual predictions within each category.

Modifying the class weights is not a very difficult task and can be easily done with a little bit of iterative tuning.

However, it is better to build the ANN first to be as accurate as possible and then use the class weights only for further tuning.

We will try to balance the accuracy of predictions with maintaining the shape of the predictions as much as possible.

With our ANN and the adjustment to the class weights, we are able to predict and categorise about 12 Death events accurately and detect 27 Deaths accurately.

As indicated earlier, another thing to watch out for with ANN is that Just like over-fitting or over-training, we can suffer from over-optimising the ANN to the validation set. It is pretty easy to achieve 80% accuracy on the validation set only for the accuracy to reduce to 73%-74% when tried against another set.

3.2.5.2.2 Functional API Let us see what we can achieve with Functional API next

```
## 9/9 - 0s - 40ms/step

## [1] ================================

## [1] The Table of Predicted values is:

##
##   0   1   2   3   4   5   6   7
## 224  25   3  12   3   1   3   4

## [1] ====================================

## [1] The table of actual values is

##
##   0   1   2   3   4   5   6   7
## 229  18   3   9   4   2   5   5

## [1] ================================

## [1] Validation accuracy is :

## [1] 0.7963636

## [1] ================================

## [1] The Confusion Matrix is

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1   2   3   4   5   6   7
##          0 206   4   0   8   2   1   2   1
##          1   6  11   3   0   1   1   1   2
##          2   2   0   0   0   0   0   0   1
##          3   8   2   0   1   0   0   1   0
##          4   2   0   0   0   1   0   0   0
##          5   1   0   0   0   0   0   0   0
##          6   1   1   0   0   0   0   0   1
##          7   3   0   0   0   0   0   1   0
##
## Overall Statistics
##
##                Accuracy : 0.7964
##                  95% CI : (0.7439, 0.8424)
##     No Information Rate : 0.8327
##     P-Value [Acc > NIR] : 0.9522
##
##                   Kappa : 0.3506
```

```
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.8996  0.61111  0.00000 0.111111 0.250000 0.000000
## Specificity            0.6087  0.94553  0.98897 0.958647 0.992620 0.996337
## Pos Pred Value         0.9196  0.44000  0.00000 0.083333 0.333333 0.000000
## Neg Pred Value         0.5490  0.97200  0.98897 0.969582 0.988971 0.992701
## Prevalence             0.8327  0.06545  0.01091 0.032727 0.014545 0.007273
## Detection Rate         0.7491  0.04000  0.00000 0.003636 0.003636 0.000000
## Detection Prevalence   0.8145  0.09091  0.01091 0.043636 0.010909 0.003636
## Balanced Accuracy      0.7541  0.77832  0.49449 0.534879 0.621310 0.498168
##                      Class: 6 Class: 7
## Sensitivity           0.00000  0.00000
## Specificity           0.98889  0.98519
## Pos Pred Value        0.00000  0.00000
## Neg Pred Value        0.98162  0.98155
## Prevalence            0.01818  0.01818
## Detection Rate        0.00000  0.00000
## Detection Prevalence  0.01091  0.01455
## Balanced Accuracy     0.49444  0.49259

## [1] ================================

## [1] ================================

## [1] The Number of Deaths correctly categorised is :

## [1] 13

## [1] ================================

## [1] ================================

## [1] The Number of Deaths correctly detected is :

## [1] 28

## [1] ================================
```

With Functional API, we are able to accurately predict and categorise 13 deaths. We are able to detect 28 deaths accurately. This is just about marginally better than what we could do with Sequential API.

Where we have some real progress with Functional API is in the shape of the predictions. They are a lot more closer to the actual shape of predictions than XGBoost or ANN built with Sequential API.

However, we need to evaluate the performance of all against the holdout set before drawing further conclusions.

3.3 Predictions for Holdout Set

We will now perform our predictions for the Holdout sets using the same models as we did for the Cross Validation sets. With Naive Bayes and XGBoost , we just reuse the parameters that we have used in the Analysis section of this chapter.

3.3.1 Single Category for Deaths

3.3.1.1 Naive Bayes without Imputation - Binary Outcome

```
##
## ================================== Naive Bayes ==================================
##
## - Call: naive_bayes.default(x = mic_modified_train[, 2:112], y = mic_modified_train$LET_IS,      lap
## - Laplace: 1
## - Classes: 2
## - Samples: 1357
## - Features: 111
## - Conditional distributions:
##     - Bernoulli: 78
##     - Categorical: 21
##     - Poisson: 6
##     - Gaussian: 6
## - Prior probabilities:
##     - 0: 0.8423
##     - 1: 0.1577
##
## ---------------------------------------------------------------------------------
```

```
## [1] ================================
```

```
## [1] The table of predictions is
```

```
##
##   0   1
## 279  64
```

```
## [1] ================================
```

```
## [1] ================================
```

```
## [1] The table of actual values is
```

```
##
##   0   1
## 286  57
```

```
## [1] ================================
```

```
## [1] The accuracy of predictions is
```

```
## [1] 0.8221574
```

```
## [1] The confusion matrix is
```

```
## [1] ================================
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 252  27
##          1  34  30
##
##                Accuracy : 0.8222
##                  95% CI : (0.7775, 0.8611)
##     No Information Rate : 0.8338
##     P-Value [Acc > NIR] : 0.7460
##
##                   Kappa : 0.3883
##
##  Mcnemar's Test P-Value : 0.4424
##
##             Sensitivity : 0.8811
##             Specificity : 0.5263
##          Pos Pred Value : 0.9032
##          Neg Pred Value : 0.4687
##              Prevalence : 0.8338
##          Detection Rate : 0.7347
##    Detection Prevalence : 0.8134
##       Balanced Accuracy : 0.7037
##
##        'Positive' Class : 0
##
```

```
## [1] ================================
```

Naive Bayes without imputation performs lower for the holdout set than it does with the cross validation set.

The Number of deaths predicted has increased only slightly. The number of deaths missed and the number of live cases reported as deaths has increased a lot causing a much larger drop in both overall accuracy and the balanced accuracy. The Specificity has also degraded when compared to the initial Analysis.

3.3.1.2 Naive Bayes with Imputation - Binary Outcome

```
##
## ================================ Naive Bayes ==================================
##
## - Call: naive_bayes.default(x = mic_modified_train[, 2:112], y = mic_modified_train$LET_IS,      lap
## - Laplace: 1
## - Classes: 2
## - Samples: 1357
```

```
## - Features: 111
## - Conditional distributions:
##     - Bernoulli: 78
##     - Categorical: 21
##     - Poisson: 7
##     - Gaussian: 5
## - Prior probabilities:
##     - 0: 0.8423
##     - 1: 0.1577
##
## ---------------------------------------------------------------------------------
```

```
## [1] ================================
```

```
## [1] The table of predictions is:
```

```
##
##   0   1
## 271  72
```

```
## [1] ================================
```

```
## [1] ================================
```

```
## [1] The table of actual values is:
```

```
##
##   0   1
## 286  57
```

```
## [1] ================================
```

```
## [1] The accuracy of predictions is:
```

```
## [1] 0.8046647
```

```
## [1] The confusion matrix is:
```

```
## [1] ================================
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 245  26
##          1  41  31
##
##                Accuracy : 0.8047
##                  95% CI : (0.7587, 0.8453)
##     No Information Rate : 0.8338
##     P-Value [Acc > NIR] : 0.9337
```

```
##
##                   Kappa : 0.3623
##
##  Mcnemar's Test P-Value : 0.0872
##
##             Sensitivity : 0.8566
##             Specificity : 0.5439
##          Pos Pred Value : 0.9041
##          Neg Pred Value : 0.4306
##              Prevalence : 0.8338
##          Detection Rate : 0.7143
##    Detection Prevalence : 0.7901
##       Balanced Accuracy : 0.7003
##
##        'Positive' Class : 0
##
```

```
## [1] ================================
```

We can see the same behaviour with the imputed dataset. The number of deaths missed and the number of live cases reported as deaths has increased dragging down the overall accuracy and the balanced accuracy. Again the Specificity has also degraded when compared to the initial Analysis.

We can see a pattern start to emerge that is an indication of the complexity of this dataset and the difficulty of making accurate predictions.

3.3.1.3 XGBoost

Let us see if XGBoost can fare any better

```
## [1] ================================

## [1] The Details for XGBoost are

## ##### xgb.Booster
## call:
##   xgb.train(data = dtrain_final, nrounds = 200, objective = "binary:logistic",
##     verbose = 0, max_depth = 10, eta = 0.15, nthread = 8)
## # of features: 111
## # of rounds:  200

## [1] ================================

## [1] ================================

## [1] The table of predictions is

##
##   0   1
## 309  34

## [1] ================================

## [1] ================================

## [1] The table of actual values is

##
##   0   1
## 286  57

## [1] ================================

## [1] ================================

## [1] 0.9154519

## [1] The confusion matrix is

## [1] ================================

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 283  26
##          1   3  31
##
```

```
##                Accuracy : 0.9155
##                  95% CI : (0.8808, 0.9426)
##     No Information Rate : 0.8338
##     P-Value [Acc > NIR] : 8.318e-06
##
##                   Kappa : 0.6361
##
##  Mcnemar's Test P-Value : 4.402e-05
##
##             Sensitivity : 0.9895
##             Specificity : 0.5439
##          Pos Pred Value : 0.9159
##          Neg Pred Value : 0.9118
##              Prevalence : 0.8338
##          Detection Rate : 0.8251
##    Detection Prevalence : 0.9009
##       Balanced Accuracy : 0.7667
##
##        'Positive' Class : 0
##
```

```
## [1] ================================
```

XGBoost is a lot more consistent in its behaviour for predictions for the Holdout set. The overall accuracy for predictions is slightly better and the balanced accuracy is slightly lower than for the Cross Validation set. We can see some reduction in the Specificity but the results are largely consistent.

3.3.1.4 Artificial Neural Networks

3.3.1.4.1 Sequential API Let us evaluate Sequential API now

```
## 11/11 - 0s - 13ms/step
```

```
## [1] ================================
```

```
## [1] The Table of Predicted values is
```

```
##
##   0   1
## 295  48
```

```
## [1] ================================
```

```
## [1] ================================
```

```
## [1] The table of actual values is
```

```
##
##   0   1
## 286  57
```

```
## [1] ================================

## [1] ================================

## [1] "The overall accuracy is : " "0.8601"

## [1] ================================

## [1] ================================

## [1] The Confusion Matrix is

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 258  20
##          1  28  37
##
##                Accuracy : 0.8601
##                  95% CI : (0.8188, 0.895)
##     No Information Rate : 0.8338
##     P-Value [Acc > NIR] : 0.1070
##
##                   Kappa : 0.5219
##
##  Mcnemar's Test P-Value : 0.3123
##
##             Sensitivity : 0.9021
##             Specificity : 0.6491
##          Pos Pred Value : 0.9281
##          Neg Pred Value : 0.5692
##              Prevalence : 0.8338
##          Detection Rate : 0.7522
##    Detection Prevalence : 0.8105
##       Balanced Accuracy : 0.7756
##
##        'Positive' Class : 0
##

## [1] ================================

## [1] deaths detected:  37

## [1] deaths missed:  20

## [1] survival cases flagged as deaths:  28
```

With ANN built using the Keras Sequential API, we are able to achieve a decent balanced accuracy and predict a higher number of deaths more accurately.

However, our balanced accuracy is getting dragged down due to the large number of survival events incorrectly predicted as lethal events. Though at first glance it may seem that we are doing worse than XGBoost, we must remind ourselves that the biggest challenge with the MIC dataset is to accurately predict the deaths or lethal events more than the survival events and we have made some incremental progress towards predicting more deaths accurately.

3.3.1.4.2 Functional API

We evaluate Functional API next

```
## 11/11 - 0s - 17ms/step

## [1] ================================

## [1] The Table of Predicted values is

##
##   0   1
## 290  53

## [1] ================================

## [1] ================================

## [1] The table of actual values is

##
##   0   1
## 286  57

## [1] ================================

## [1] ================================

## [1] The overall accuracy is :  0.8455

## [1] ================================

## [1] ================================

## [1] The Confusion Matrix is

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 249  16
##          1  37  41
##
##                Accuracy : 0.8455
##                  95% CI : (0.8028, 0.8821)
##     No Information Rate : 0.8338
##     P-Value [Acc > NIR] : 0.31012
##
##                   Kappa : 0.5141
##
##  Mcnemar's Test P-Value : 0.00601
##
```

```
##             Sensitivity : 0.8706
##             Specificity : 0.7193
##          Pos Pred Value : 0.9396
##          Neg Pred Value : 0.5256
##              Prevalence : 0.8338
##          Detection Rate : 0.7259
##    Detection Prevalence : 0.7726
##       Balanced Accuracy : 0.7950
##
##        'Positive' Class : 0
##
```

```
## [1] ================================
```

```
## [1] deaths detected:  41
```

```
## [1] deaths missed:  16
```

```
## [1] survival cases flagged as deaths:  37
```

With ANN built using Keras Functional API, we are getting a bit better in being able to predict lethal or death events. At the same time the number of live cases being flagged as deaths has increased quite dramatically. The Overall Accuracy and the Balanced Accuracy are quite good. We have the best Specificity as yet. Sensitivity has suffered a bit.

As I have said earlier, this ANN model is not a true reflection of the capabilities of ANN. This is more a reflection of my own capabilities in building ANN. This project is the first for me to try ANN and I have had to learn on my feet over the last month or so by looking at the Keras documentation. I strongly believe that ANN models can be built to predict with much better overall and balanced accuracies.

3.3.2 Multiple Categories for Deaths

Let us now look at the predictions for our holdout set for Multiple categories of deaths. We start with XGBoost

3.3.2.1 XGBoost

```
## [1] ====================================
```

```
## [1] The details for XGBoost are
```

```
## ##### xgb.Booster
## call:
##   xgb.train(params = list(num_class = 8, booster = "gbtree"), data = dtrain_categorical_final,
##     nrounds = 200, objective = "multi:softmax", verbose = 0,
##     max_depth = 10, eta = 0.15, nthread = 8)
## # of features: 111
## # of rounds:  200
```

```
## [1] ====================================
```

```
## [1] ====================================
```

```
## [1] The table of predictions is
```

```
##
##   0   1   2   3   4   5   6
## 316  17   2   3   2   1   2
```

```
## [1] "===================================="
```

```
## [1] ====================================
```

```
## [1] The table of actual values is
```

```
##
##   0   1   2   3   4   5   6   7
## 286  22   4  11   5   3   6   6
```

```
## [1] ====================================
```

```
## [1] The accuracy of predictions is
```

```
## [1] 0.8862974
```

```
## [1] ====================================
```

```
## [1] The confusion matrix is
```

```
## [1] ====================================

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1   2   3   4   5   6   7
##          0 286   4   3   8   3   3   5   4
##          1   0  15   1   0   0   0   0   1
##          2   0   2   0   0   0   0   0   0
##          3   0   0   0   2   0   0   1   0
##          4   0   0   0   1   1   0   0   0
##          5   0   0   0   0   1   0   0   0
##          6   0   1   0   0   0   0   0   1
##          7   0   0   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.8863
##                  95% CI : (0.8479, 0.9179)
##     No Information Rate : 0.8338
##     P-Value [Acc > NIR] : 0.004115
##
##                   Kappa : 0.5015
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            1.0000  0.68182 0.000000 0.181818 0.200000 0.000000
## Specificity            0.4737  0.99377 0.994100 0.996988 0.997041 0.997059
## Pos Pred Value         0.9051  0.88235 0.000000 0.666667 0.500000 0.000000
## Neg Pred Value         1.0000  0.97853 0.988270 0.973529 0.988270 0.991228
## Prevalence             0.8338  0.06414 0.011662 0.032070 0.014577 0.008746
## Detection Rate         0.8338  0.04373 0.000000 0.005831 0.002915 0.000000
## Detection Prevalence   0.9213  0.04956 0.005831 0.008746 0.005831 0.002915
## Balanced Accuracy      0.7368  0.83779 0.497050 0.589403 0.598521 0.498529
##                      Class: 6 Class: 7
## Sensitivity          0.000000  0.00000
## Specificity          0.994065  1.00000
## Pos Pred Value       0.000000      NaN
## Neg Pred Value       0.982405  0.98251
## Prevalence           0.017493  0.01749
## Detection Rate       0.000000  0.00000
## Detection Prevalence 0.005831  0.00000
## Balanced Accuracy    0.497033  0.50000

## [1] ====================================

## [1] ================================

## [1] The Number of Deaths correctly categorised is :

## [1] 18
```

```
## [1] ================================

## [1] ================================

## [1] The Number of Deaths correctly detected is :

## [1] 27

## [1] ================================

## [1] ================================

## [1] The Number of Live cases flagged as Deaths is :

## [1] 0

## [1] ================================
```

As usual XGBoost is a consistent performer. XGboost is able to correctly detect and categorise 18 deaths while being able to correctly detect 27 deaths. The overall accuracy is extremely good but comes at the huge cost of predicting more than half of the deaths as live cases, which is not what we want in this particular case.

However, the shape of our predictions has improved quite a bit from the initial analysis.

3.3.2.2 Artificial Neural Networks

We now turn our attention to ANN built using Sequential API

3.3.2.2.1 Sequential API

```
## 11/11 - 0s - 18ms/step

## [1] ================================

## [1] The Table of Predicted values is

##
##   0   1   2   3   4   5   6   7
## 256  30   4  23  11  10   6   3

## [1] ================================

## [1] ====================================

## [1] The table of actual values is

##
##   0   1   2   3   4   5   6   7
## 286  22   4  11   5   3   6   6

## [1] ================================

## [1] ================================

## [1] The overall accuracy is :

## [1] 0.7696793

## [1] ================================

## [1] The Confusion Matrix is

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1   2   3   4   5   6   7
##          0 243   5   1   2   0   0   2   3
##          1  15  10   1   0   1   0   2   1
##          2   2   1   0   0   1   0   0   0
##          3   9   5   0   8   1   0   0   0
##          4   6   0   2   0   1   0   1   1
##          5   4   1   0   1   1   2   1   0
##          6   5   0   0   0   0   0   0   1
##          7   2   0   0   0   0   1   0   0
##
## Overall Statistics
```

```
##
##                Accuracy : 0.7697
##                  95% CI : (0.7214, 0.8132)
##     No Information Rate : 0.8338
##     P-Value [Acc > NIR] : 0.9991
##
##                   Kappa : 0.3751
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.8497  0.45455  0.00000  0.72727 0.200000 0.666667
## Specificity            0.7719  0.93769  0.98820  0.95482 0.970414 0.976471
## Pos Pred Value         0.9492  0.33333  0.00000  0.34783 0.090909 0.200000
## Neg Pred Value         0.5057  0.96166  0.98820  0.99063 0.987952 0.996997
## Prevalence             0.8338  0.06414  0.01166  0.03207 0.014577 0.008746
## Detection Rate         0.7085  0.02915  0.00000  0.02332 0.002915 0.005831
## Detection Prevalence   0.7464  0.08746  0.01166  0.06706 0.032070 0.029155
## Balanced Accuracy      0.8108  0.69612  0.49410  0.84105 0.585207 0.821569
##                      Class: 6 Class: 7
## Sensitivity           0.00000 0.000000
## Specificity           0.98220 0.991098
## Pos Pred Value        0.00000 0.000000
## Neg Pred Value        0.98220 0.982353
## Prevalence            0.01749 0.017493
## Detection Rate        0.00000 0.000000
## Detection Prevalence  0.01749 0.008746
## Balanced Accuracy     0.49110 0.495549

## [1] "================================="

## [1] =================================

## [1] The Number of Deaths correctly categorised is :

## [1] 21

## [1] =================================

## [1] =================================

## [1] The Number of Deaths correctly detected is :

## [1] 44

## [1] =================================

## [1] =================================

## [1] The Number of Live cases flagged as Deaths is :
```

```
## [1] 43

## [1] ================================
```

The Results are largely as expected. The results are a lot worse than they are for binary predictions. We are not able to predict across the whole range too. The prediction accuracy for the Holdout set is again a lot worse than that for the cross validation set reflecting the complexity of the dataset.

However, we have made some progress in the number of deaths correctly detected and categorised as well as the number of deaths detected. This has come at the cost of flagging many live cases as deaths.

3.3.2.2.2 Functional API Let us now check how ANN built using Functional API perform

```
## 11/11 - 0s - 34ms/step

## [1] ================================

## [1] The Table of Predicted values is

##
##   0   1   2   3   4   5   6   7
## 278  28   4  14   3   6   3   7

## [1] ================================

## [1] ================================

## [1] The Table of actual values is

##
##   0   1   2   3   4   5   6   7
## 286  22   4  11   5   3   6   6

## [1] ================================

## [1] The overall accuracy is

## [1] 0.7959184

## [1] ================================

## [1] The Confusion Matrix is

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1   2   3   4   5   6   7
##          0 256   7   2   5   1   1   2   4
##          1  11  12   1   1   1   0   2   0
##          2   3   0   0   0   0   0   0   1
```

```
##          3   4   3   1   4   0   1   1   0
##          4   3   0   0   0   0   0   0   0
##          5   4   0   0   0   0   1   1   0
##          6   1   0   0   0   1   0   0   1
##          7   4   0   0   1   2   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.7959
##                  95% CI : (0.7493, 0.8373)
##     No Information Rate : 0.8338
##     P-Value [Acc > NIR] : 0.9724
##
##                   Kappa : 0.3556
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.8951  0.54545  0.00000  0.36364 0.000000 0.333333
## Specificity            0.6140  0.95016  0.98820  0.96988 0.991124 0.985294
## Pos Pred Value         0.9209  0.42857  0.00000  0.28571 0.000000 0.166667
## Neg Pred Value         0.5385  0.96825  0.98820  0.97872 0.985294 0.994065
## Prevalence             0.8338  0.06414  0.01166  0.03207 0.014577 0.008746
## Detection Rate         0.7464  0.03499  0.00000  0.01166 0.000000 0.002915
## Detection Prevalence   0.8105  0.08163  0.01166  0.04082 0.008746 0.017493
## Balanced Accuracy      0.7546  0.74781  0.49410  0.66676 0.495562 0.659314
##                      Class: 6 Class: 7
## Sensitivity          0.000000  0.00000
## Specificity          0.991098  0.97923
## Pos Pred Value       0.000000  0.00000
## Neg Pred Value       0.982353  0.98214
## Prevalence           0.017493  0.01749
## Detection Rate       0.000000  0.00000
## Detection Prevalence 0.008746  0.02041
## Balanced Accuracy    0.495549  0.48961

## [1] ================================

## [1] ================================

## [1] The Number of Deaths correctly categorised is :

## [1] 17

## [1] ================================

## [1] ================================

## [1] The Number of Deaths correctly detected is :

## [1] 35
```

```
## [1] ================================

## [1] ================================

## [1] The Number of Live cases flagged as Deaths is :

## [1] 30

## [1] ================================
```

The ANN built using the Functional API offer a middle-ground between the predictions from XGBoost and ANN built with the Sequential API. They offer better accuracies than XGBoost in the number of deaths detected. They also offer lower number of number of lives cases flagged as deaths when compared to ANN built with Sequential API.

Where ANN using Functional API really provide us with some advantage is in the shape of the predictions, they offer much better shape to the predictions that XGBoost or ANN built using Sequential API.

3.4 Results & Inference

The MIC dataset is a lot more complex and has more features when compared to the HFP dataset. It also has a much smaller number of observations to work with, making it more difficult to extract patterns for predictions.

We have also seen that the predictions for the Validation set and the Holdout sets also vary by much.

3.4.1 Additional Observations

Let us record some additional observations that apply to both the binary outcome and the multi category outcome.

1. The choice of the validation and holdout sets is more providential than intentional as I have used the same Random Number Generator seed (1024) throughout for keeping the code and scheme simple and consistent. However, the holdout set that results from using this seed is actually quite a difficult one to make predictions against.
2. **It is easy to modify the seed and choose different validation and holdout sets. For those who are curious. I can recommend setting the seed to 4096 for partitioning the MIC dataset between Training and Testing sets. You can see a dramatic improvement in the Predictions offered by some of the Algorithms like ANN and Naive Bayes.**
3. With ANN, depending on the way the ANN is built, the predictions can vary a lot when trained using different sets. Another situation to watch out for with ANN is that when they are very well optimised for the Validation set, they often perform poorly against the Test sets. The behaviour is similar to what we can see with over-training or over-fitting.
4. When ANN are built using unordered factors as compared to using Ordinals (ordered factors), they can be built to offer very high accuracies but unfortunately they lack consistency in their predictions, and the predictions between the validation and holdout sets vary a lot. Using Ordinals makes the predictions a lot more consistent.
5. XGBoost and ANN have a lot of tuning parameters that can greatly enhance their accuracy. Whatever I have built are just the most basic for purposes of comparison. With the right amount of skill and time, they can be tuned to offer far more accurate predictions.

Let us summarise and review our analysis so far

3.4.2 Single Category for Deaths

```
## [1] The Survival Rate for patients on this dataset is:
## [2] 0.8406
```

Table 7: Myocardial Infarction Complications - Results Summary - Binary Outcome

Model	overall accuracy	balanced accuracy	deaths detected	deaths missed	live flagged
Naive Bayes without Imputation	0.8222	0.7037	30	27	34
Naive Bayes with Imputation	0.8047	0.7003	31	26	41
XGBoost	0.9155	0.7667	31	26	3
ANN Sequential API	0.8601	0.7756	37	20	28
ANN Functional API	0.8455	0.7950	41	16	37

When it comes to predictions with a simple binary outcome, we can see that XGBoost does a grand job in terms of overall accuracy and also has very good balanced accuracy. It does a pretty good job in terms of

the deaths detected. For anybody who is looking for a solution that is quick and requires minimal effort to configure and run, XGBoost is the right choice.

ANN do a better job in terms of balanced accuracy. As stated before many times, the ANN that I have built are just very basic and rudimentary. With the right knowledge and skills, ANN can be built to be far more accurate.

XGBoost also has many tuning parameters and can provide further improvements in accuracy. as well

3.4.3 Multiple Categories for Deaths

Table 8: MIC - Results Summary - Categorical Outcome

Model	Accuracy	Deaths Accurately Categorised	Deaths Detected	Deaths Missed	Live Cases Flagged
XGBoost	0.8863	18	27	30	0
ANN Sequential API	0.7697	21	44	13	43
ANN Functional API	0.7959	17	35	22	30

3.4.3.1 Distributions of Predictions Let us look at the distribution of predictions between the various algorithms

Table 9: MIC - Table of distributions

LET_IS	Actual Values	XGBoost	ANN Sequential API	ANN Functional API
0	286	316	256	278
1	22	17	30	28
2	4	2	4	4
3	11	3	23	14
4	5	2	11	3
5	3	1	10	6
6	6	2	6	3
7	6	0	3	7

Table 10: MIC - Table of Accurate Predictions

LET_IS	Actual Values	XGBoost	ANN Sequential API	ANN Functional API
0	286	286	243	256
1	22	15	10	12
2	4	0	0	0
3	11	2	8	4
4	5	1	1	0
5	3	0	2	1
6	6	0	0	0
7	6	0	0	0

With Multiple Categories for Deaths too, the inference is somewhat similar to the Single Category. XGBoost is a good choice for somebody who is looking for a quick solution that is easy on the effort required. However the shape of predictions is poor and XGBoost is more biased towards predicting more deaths as live cases which is counter productive to our efforts to predict more deaths accurately. However, we are using XGBoost only with some basic tuning. With more advanced tuning XGBoost can do quite well

ANN require a lot more programming skill and effort, but can provide better accuracy if built and tuned properly.

4 Diabetes 130-US Hospitals for Years 1999-2008

We will pick up the largest dataset at the end.

4.1 Dataset Introduction

4.1.1 List of Attributes/Features

The list of Attributes/Features is very well described in the Research Article available at https://doi.org/10.1155/2014/781670.

We present a slightly modified version of the table available in the article below, which has updates to the missing values to be aligned with the dataset under analysis. Another important feature of the raw data is that the missing values are all represented using a "?" mark, which makes it very easy to identify them.

Table 11: D130 Attributes/Features - Columns 1-24

Col ID	Feature name	Type	Description and values	% missing
1	encounter_id	Numeric / Integer	Unique identifier of an encounter	0
2	patient_nbr	Numeric / Integer	Unique identifier of a patient	0
3	race	Categorical	Values: Caucasian, Asian, African American, Hispanic, and other	2.33
4	gender	Categorical	Values: male, female, and unknown/invalid	0
5	age	Categorical	Grouped in 10-year intervals: [0, 10), [10, 20),..., [90, 100)	0
6	weight	Numeric	Weight in pounds.	96.86
7	admission_type_id	Categorical	Integer identifier corresponding to 9 distinct values, for example, emergency, urgent, elective, newborn, and not available	0
8	discharge_disposition_id	Categorical	Integer identifier corresponding to 29 distinct values, for example, discharged to home, expired, and not available	0
9	admission_source_id	Categorical	Integer identifier corresponding to 21 distinct values, for example, physician referral, emergency room, and transfer from a hospital	0
10	time_in_hospital	Integer	Integer number of days between admission and discharge	0
11	payer_code	Categorical	Integer identifier corresponding to 23 distinct values, for example, Blue Cross/Blue Shield, Medicare, and self-pay	39.56

12	medical_specialty	Categorical	Integer identifier of a specialty of the admitting physician, corresponding to 84 distinct values, for example, cardiology, internal medicine, family/general practice, and surgeon	49.08
13	num_lab_procedures	Integer	Number of lab tests performed during the encounter	0
14	num_procedures	Integer	Number of procedures (other than lab tests) performed during the encounter	0
15	num_medications	Integer	Number of distinct generic names administered during the encounter	0
16	number_outpatient	Integer	Number of outpatient visits of the patient in the year preceding the encounter	0
17	number_emergency	Integer	Number of emergency visits of the patient in the year preceding the encounter	0
18	number_inpatient	Integer	Number of inpatient visits of the patient in the year preceding the encounter	0
19	diag_1	Categorical	The primary diagnosis (coded as first three digits of ICD9); 717 distinct values	0.02
20	diag_2	Categorical	Secondary diagnosis (coded as first three digits of ICD9); 749 distinct values	0.35
21	diag_3	Categorical	Additional secondary diagnosis (coded as first three digits of ICD9); 790 distinct values	1.4
22	number_diagnoses	Integer	Number of diagnoses entered to the system	0
23	max_glu_serum	Categorical	Indicates the range of the result or if the test was not taken. Values: >200, >300, normal, and none if not measured	0
24	A1Cresult	Categorical	Indicates the range of the result or if the test was not taken. Values: >8 if the result was greater than 8%, >7 if the result was greater than 7% but less than 8%, normal if the result was less than 7%, and none if not measured.	0

Columns 25-47
Description and values
23 features for medications -
The feature indicates whether the drug was prescribed or there was a change in the dosage.
Values: “up” if the dosage was increased during the encounter,
“down” if the dosage was decreased,
“steady” if the dosage did not change,
and “no” if the drug was not prescribed

Table 13: D130 Attributes/Features - Columns 25-47

Col ID	Feature name	Type	Description and values	% missing
25	metformin	Categorical	See note above	0
26	repaglinide	Categorical	See note above	0
27	nateglinide	Categorical	See note above	0
28	chlorpropamide	Categorical	See note above	0
29	glimepiride	Categorical	See note above	0
30	acetohexamide	Categorical	See note above	0
31	glipizide	Categorical	See note above	0
32	glyburide	Categorical	See note above	0
33	tolbutamide	Categorical	See note above	0
34	pioglitazone	Categorical	See note above	0
35	rosiglitazone	Categorical	See note above	0
36	acarbose	Categorical	See note above	0
37	miglitol	Categorical	See note above	0
38	troglitazone	Categorical	See note above	0
39	tolazamide	Categorical	See note above	0
40	examide	Categorical	See note above	0
41	citoglipton	Categorical	See note above	0
42	insulin	Categorical	See note above	0
43	glyburide-metformin	Categorical	See note above	0
44	glipizide-metformin	Categorical	See note above	0
45	glimepiride-pioglitazone	Categorical	See note above	0
46	metformin-rosiglitazone	Categorical	See note above	0
47	metformin-pioglitazone	Categorical	See note above	0

Table 14: D130 Attributes/Features - Columns 48-49

Col ID	Feature name	Type	Description and values	% missing
48	change	Categorical	Indicates if there was a change in diabetic medications (either dosage or generic name). Values: change and no change	0
49	diabetesMed	Categorical	Indicates if there was any diabetic medication prescribed. Values: yes and no	0

Table 15: D130 Outcome/Prediction - Column 50

Col ID	Target	Type	Description and values	% missing
50	readmitted	Categorical	Days to inpatient readmission. 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.	0

Let us have a quick look at some of the Attributes and see what we can make of them at first glance.

1. encounter_id - It is a unique value for every interaction and is the unique key for each observation in the Dataset. There are as many unique "encounter_id" values as there are rows in the dataset i.e 101766.
2. patient_nbr- It is a unique value for each Patient and can help us track the number of encounter_id associated with each patient. The total number of unique "patient_nbr" values is 71518 which is about 70.28% of the total number of rows. We shall treat the "patient_nbr" as a factor type variable.
3. race - It is a categorical variable with values Caucasian, Asian, African American, Hispanic, and other. About 2.33% of the values are missing. It does not make much sense to try to impute values into the missing rows based on the data from the other columns. We can choose to either include the missing values under the "other" category or leave it as it is. We will choose the latter option so not to taint the original data. Remember that in this dataset, unknown values are represented by a "?".
4. Weight - Though being overweight is a very important indicator for developing Diabetes and progression of the disease, we will also need some indication of whether the person is overweight. Though Race, Gender and Age can provide some indications, these are too broad to be able to guess whether a person is overweight or not when missing almost 97% of the values. Too much of the data is missing and it is unlikely that we can impute something based on the data from the other columns that is sensible.
5. payer_code - About 39.56% of the values are missing. It is likely that this column actually codes some important data about when the patient decides to seek medical attention, Again imputation using the other columns would possibly be erroneous.
6. medical_specialty - About 49.08% of the values are missing. Again it is very likely that this column codes important data about the initial diagnosis and progression of the disease before the patients (or their caregivers) sought specific medical attention for diabetes. It will be unlikely to impute missing data.
7. diag_1, diag_2 and diag_3 - There are a very small numbers of missing values, though there are 717 distinct possibilities for diag_1, 749 for diag_2 and 790 for diag_3 respectively, Since these are all based on ICD9 codes, any imputation using the other columns would possibly be erroneous.

4.1.2 Problem to Solve

We need to be able to predict the days to inpatient readmission. The values are

1. "<30" if the patient was readmitted in less than 30 days
2. ">30" if the patient was readmitted in more than 30 days, and
3. "No" for no record of readmission.

4.1.3 Important Details

Next, we explore some very important details about the dataset that are not available from the UCI Website but can be understood by reading through the exemplary work done by the researchers, which is available in their publication[2].

Here is an extract from the paper with the authors' comments reproduced verbatim but organised as a ordered list for ease of reading.

---

1. Criteria for Inclusion.

   (1) It is an inpatient encounter (a hospital admission).
   (2) It is a "diabetic" encounter, that is, one during which any kind of diabetes was entered to the system as a diagnosis.
   (3) The length of stay was at least 1 day and at most 14 days.
   (4) Laboratory tests were performed during the encounter.
   (5) Medications were administered during the encounter.

2. Criteria 3-4 were applied to remove admissions for procedures and so forth, which were of less than 23 hours of duration and in which changes in diabetes management were less likely to have occurred.

3. **It should be noted that the diabetic encounters are not all encounters of diabetic patients but rather only these encounters where diabetes was coded as an existing health condition.**

4. 101,766 encounters were identified to fulfil all of the above five inclusion criteria and were used in further analysis. Attribute/feature selection was performed by our clinical experts and only attributes that were potentially associated with the diabetic condition or management were retained.

5. Since we are primarily interested in factors that lead to early readmission, we defined the readmission attribute (out- come) as having two values: "readmitted," if the patient was readmitted within 30 days of discharge or "otherwise," which covers both readmission after 30 days and no readmission at all.

6. The values of the readmission attribute were determined by examination of all patient records in the database to determine the first inpatient visit after discharge. Note that 30 days was chosen based on criteria often used by funding agencies.

7. Haemoglobin A1c (HbA1c) is an important measure of glucose control, which is widely applied to measure performance of diabetes care.

   1. The measurement of HbA1c at the time of hospital admission offers a unique opportunity to assess the efficacy of current therapy and to make changes in that therapy if indicated (e.g., HbA1c $>$ 8.0% on current regimen). We considered the possibility that if an HbA1c test result was available from a measurement (outpatient or inpatient) done within three months prior to the sentinel admission, the test might not be repeated. In these cases (0.1% of the total), we used the measurement available from the previous visit. In all other cases, measurement of HbA1c was performed at the time of hospital admission.

2. We examined both the frequency of HbA1c test ordering and the response to its result, which we defined as a change in diabetic medications. By a "change of medication" we understand any dosage change (increase or reduction) as well as change to a drug with a different generic name, for example, a change of the type of insulin or an introduction of a new drug. The database contains detailed information about dosage but is restricted only to medications administered during the encounter. It was not possible to track any preadmission and discharge medications.
3. We considered 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% with no change in diabetic medications, and (4) HbA1c performed, result is greater than 8%, and diabetic medication was changed.

As a reading of the Research Article would indicate, the primary emphasis of the article is on the efficacy of using HbA1c measurements for managing diabetes and its progression, through changes in treatment.

4.2 Analysis

4.2.1 Dataset preparation

We will download the dataset from the UCI ML repository. As indicated in the Introduction chapter, the dataset is available in CSV format enclosed in a zip file. It can easily be unzipped and imported into a data frame for further processing.

We will create Lists of Continuous, Discrete, Categorical and Binary features so that it is easier to process them later.

We will also create feature sets or groupings of features that we can use with the Keras Functional API. The feature set groupings are done with very rudimentary knowledge of the features based on their descriptions and the availability of time. As indicated already, we can greatly enhance the accuracy of the predictions with the services of a Subject Matter Expert (SME) in CVD who can help us group features that can be combined together to offer the best possible predictions.

We will assign the right types to each variable. We will configure Binary and Discrete variables as factors, Discrete variables as Integers and Continuous variables as Numerics (Float). There is only one Continuous variable which is "weight" and it is not relevant to our analysis. we will document it for the sake of completeness.

Let us get a view of the ratios of the outcomes in the dataset.

```
## [1] The structure of our d130 as imported is :

## [1] ================================

## 'data.frame':    101766 obs. of  50 variables:
##  $ encounter_id            : int  2278392 149190 64410 500364 16680 35754 55842 63768 12522 15738 ..
##  $ patient_nbr             : int  8222157 55629189 86047875 82442376 42519267 82637451 84259809 11488
##  $ race                    : chr  "Caucasian" "Caucasian" "AfricanAmerican" "Caucasian" ...
##  $ gender                  : chr  "Female" "Female" "Female" "Male" ...
##  $ age                     : chr  "[0-10)" "[10-20)" "[20-30)" "[30-40)" ...
##  $ weight                  : chr  "?" "?" "?" "?" ...
##  $ admission_type_id       : int  6 1 1 1 1 2 3 1 2 3 ...
##  $ discharge_disposition_id: int  25 1 1 1 1 1 1 1 1 3 ...
##  $ admission_source_id     : int  1 7 7 7 7 2 2 7 4 4 ...
##  $ time_in_hospital        : int  1 3 2 2 1 3 4 5 13 12 ...
##  $ payer_code              : chr  "?" "?" "?" "?" ...
##  $ medical_specialty       : chr  "Pediatrics-Endocrinology" "?" "?" "?" ...
##  $ num_lab_procedures      : int  41 59 11 44 51 31 70 73 68 33 ...
##  $ num_procedures          : int  0 0 5 1 0 6 1 0 2 3 ...
##  $ num_medications         : int  1 18 13 16 8 16 21 12 28 18 ...
##  $ number_outpatient       : int  0 0 2 0 0 0 0 0 0 0 ...
##  $ number_emergency        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ number_inpatient        : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ diag_1                  : chr  "250.83" "276" "648" "8" ...
##  $ diag_2                  : chr  "?" "250.01" "250" "250.43" ...
##  $ diag_3                  : chr  "?" "255" "V27" "403" ...
##  $ number_diagnoses        : int  1 9 6 7 5 9 7 8 8 8 ...
##  $ max_glu_serum           : chr  "None" "None" "None" "None" ...
##  $ A1Cresult               : chr  "None" "None" "None" "None" ...
##  $ metformin               : chr  "No" "No" "No" "No" ...
##  $ repaglinide             : chr  "No" "No" "No" "No" ...
```

```
##  $ nateglinide             : chr  "No" "No" "No" "No" ...
##  $ chlorpropamide          : chr  "No" "No" "No" "No" ...
##  $ glimepiride             : chr  "No" "No" "No" "No" ...
##  $ acetohexamide           : chr  "No" "No" "No" "No" ...
##  $ glipizide               : chr  "No" "No" "Steady" "No" ...
##  $ glyburide               : chr  "No" "No" "No" "No" ...
##  $ tolbutamide             : chr  "No" "No" "No" "No" ...
##  $ pioglitazone            : chr  "No" "No" "No" "No" ...
##  $ rosiglitazone           : chr  "No" "No" "No" "No" ...
##  $ acarbose                : chr  "No" "No" "No" "No" ...
##  $ miglitol                : chr  "No" "No" "No" "No" ...
##  $ troglitazone            : chr  "No" "No" "No" "No" ...
##  $ tolazamide              : chr  "No" "No" "No" "No" ...
##  $ examide                 : chr  "No" "No" "No" "No" ...
##  $ citoglipton             : chr  "No" "No" "No" "No" ...
##  $ insulin                 : chr  "No" "Up" "No" "Up" ...
##  $ glyburide.metformin     : chr  "No" "No" "No" "No" ...
##  $ glipizide.metformin     : chr  "No" "No" "No" "No" ...
##  $ glimepiride.pioglitazone: chr  "No" "No" "No" "No" ...
##  $ metformin.rosiglitazone : chr  "No" "No" "No" "No" ...
##  $ metformin.pioglitazone  : chr  "No" "No" "No" "No" ...
##  $ change                  : chr  "No" "Ch" "No" "Ch" ...
##  $ diabetesMed             : chr  "No" "Yes" "Yes" "Yes" ...
##  $ readmitted              : chr  "NO" ">30" "NO" "NO" ...

## [1] ================================

## [1] ================================

## [1] The Table of Outcomes is

##
##   <30   >30    NO
## 11357 35545 54864

## [1] Ratio of NO : 0.5391

## [1] Ratio of >30 : 0.3493

## [1] Ratio of <30  0.1116

## [1] ================================
```

For each Algorithm, the variable types required for processing are different and we need to be mindful of the same.

Before we begin with the work of analysing the dataset using various ML Algorithms, we will partition the dataset to create sets for Training and Testing. We will further partition the Training set into Training CV and Testing CV sets.

For input into Naive Bayes, we will configure categorical and binary variables as factors and discrete variables as integers.

4.2.2 Naive Bayes

Let us try with Naive Bayes to get a sense of where we can get with our predictions. As with the MIC dataset, the choice of Naive Bayes is because :-

1. It can handle missing values.
2. It can handle large numbers of factors.

Before we process the data using Naive Bayes, we will replace all missing values represented by "?" in the imported dataset with NA which is the internal representation for missing values in R.

Naive Bayes is not very complex to tune and has a few parameters that can be tuned for optimal predictions between and within the classes. Please have a look at the Naive Bayes section from the Heart Failure Prediction (HFP) chapter for more details about tuning Naive Bayes.

We will use Poisson distribution for all Discrete variables. The only Continuous variable is "weight" which we can disregard. We will disable Laplace smoothing so that we can get greater accurate predictions in the "<30" class.

4.2.2.1 Categorical outcome

```
## [1] The Naive Bayes Summary is :

## [1] ================================

##
## ================================ Naive Bayes ================================
##
## - Call: naive_bayes.default(x = d130_modified_cv_train_set[, 2:47], y = d130_modified_cv_train_set$re
## - Laplace: 0
## - Classes: 3
## - Samples: 65128
## - Features: 46
## - Conditional distributions:
##     - Bernoulli: 7
##     - Categorical: 30
##     - Poisson: 8
##     - Gaussian: 1
## - Prior probabilities:
##     - <30: 0.1116
##     - >30: 0.3493
##     - NO: 0.5391
##
## ------------------------------------------------------------------------------

## [1] ================================

## [1] ================================

## [1] The Table of predictions is :

##
##  <30  >30   NO
## 2558 3987 9739
```

```
## [1] ================================
```

```
## [1] The Table of actual values is :
```

```
##
##  <30  >30   NO
## 1817 5688 8779
```

```
## [1] ================================
```

```
## [1] The Overall Accuracy is :
```

```
## [1] 0.5486981
```

```
## [1] ================================
```

```
## [1] The confusion matrix is :
```

```
## [1] ================================
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  <30  >30   NO
##        <30  555 1271  732
##        >30  594 1863 1530
##        NO   668 2554 6517
##
## Overall Statistics
##
##                Accuracy : 0.5487
##                  95% CI : (0.541, 0.5564)
##     No Information Rate : 0.5391
##     P-Value [Acc > NIR] : 0.007228
##
##                   Kappa : 0.2145
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: <30 Class: >30 Class: NO
## Sensitivity             0.30545     0.3275    0.7423
## Specificity             0.86155     0.7995    0.5707
## Pos Pred Value          0.21697     0.4673    0.6692
## Neg Pred Value          0.90806     0.6889    0.6544
## Prevalence              0.11158     0.3493    0.5391
## Detection Rate          0.03408     0.1144    0.4002
## Detection Prevalence    0.15709     0.2448    0.5981
## Balanced Accuracy       0.58350     0.5635    0.6565
```

```
## [1] ================================
```

We have some decent shape to the Table of Predictions with our chosen parameters. Our predictions within classes are still quite poor. Again by shape we mean the overall number of predictions for each class at the level of the class.

*Note: We can achieve much higher overall accuracy by configuring the discrete variables as being Gaussian. However this leads to much poorer predictions for the "<30" and ">30" classes.

By changing the amount of Laplace smoothing by 1 and 2, we can see changes in the distributions with more number of accurate predictions being made for the ">30" and "NO" classes

Another quirk that we can observe is that by retaining the "?" as the representation for the missing values, Naive Bayes can produce slightly better results. The behaviour is possibly anomalous but the gains are very small.

The other peculiarity that we can observe with the dataset is that the categorical features provide very little increments in increasing the overall accuracy. However, they do serve the important purpose of making more accurate predictions within the "<30" and ">30" classes.

4.2.2.2 Binary outcome Let us see if we can do better with a Binary Classifier, For the "readmitted" field, we will retain the "NO" value as-is. We will consolidate the ">30" and "<30" values into a much simpler "YES".

It is worth noting at this point that the original research article consolidates the outcome for readmission in >30 days and NO readmission into a single category. This is different from our scheme for distinguishing between Readmission and NO Readmission.

For Binary predictions, we will enable Laplace smoothing with a value of 1. This is to keep the shape of the predictions from becoming too distorted.

```
## [1] The Naive Bayes Summary is :

## [1] ================================

##
## ================================ Naive Bayes ================================
##
## - Call: naive_bayes.default(x = d130_modified_cv_train_set_binary[, 2:49],      y = d130_modified_cv
## - Laplace: 1
## - Classes: 2
## - Samples: 65128
## - Features: 48
## - Conditional distributions:
##     - Bernoulli: 9
##     - Categorical: 30
##     - Poisson: 8
##     - Gaussian: 1
## - Prior probabilities:
##     - NO: 0.5391
##     - YES: 0.4609
##
## ----------------------------------------------------------------------------

## [1] ================================

## [1] ================================

## [1] The Table of predictions is :

##
##   NO  YES
## 8930 7354

## [1] ================================

## [1] The Table of actual values is :

##
##   NO  YES
## 8779 7505

## [1] ================================
```

```
## [1] The Overall Accuracy is :

## [1] 0.6301277

## [1] ================================

## [1] The confusion matrix is :

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   NO  YES
##        NO  5843 3087
##        YES 2936 4418
##
##                Accuracy : 0.6301
##                  95% CI : (0.6227, 0.6375)
##     No Information Rate : 0.5391
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.2546
##
##  Mcnemar's Test P-Value : 0.05326
##
##             Sensitivity : 0.6656
##             Specificity : 0.5887
##          Pos Pred Value : 0.6543
##          Neg Pred Value : 0.6008
##              Prevalence : 0.5391
##          Detection Rate : 0.3588
##    Detection Prevalence : 0.5484
##       Balanced Accuracy : 0.6271
##
##        'Positive' Class : NO
##

## [1] ================================
```

As Expected, our Overall Accuracy has gotten a little better. Tuning the parameters helps us predict "readmitted" = "YES" more frequently. The Shape of predictions is quite good, however both Sensitivity and Specificity are quite poor.

4.2.3 Random Forest

Let us explore Random Forest next. Since the Random Forest implementation in R cannot handle a lot of factors, we will retain the type of Categorical and Binary variables as Characters from the original dataset.

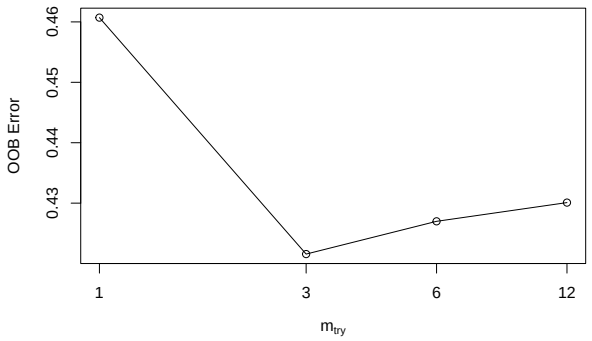
As Random Forest cannot manage missing values like Naive Bayes can, we will instead

1. Zero out the column related to weight
2. We will retain the "?" to represent missing values in payer_code, medical_specialty and other columns

```
## [1] ====================================
```

```
## [1] The Random Forest mtry values and error rates are
```

```
## mtry = 6  OOB error = 42.7%
## Searching left ...
## mtry = 12    OOB error = 43.01%
## -0.007263835 1e-05
## Searching right ...
## mtry = 3     OOB error = 42.16%
## 0.01269373 1e-05
## mtry = 1     OOB error = 46.07%
## -0.09283945 1e-05
```




```
## [1] ====================================
```

```
## [1]  Details for Random Forest for the d130 Dataset are:
```

```
## [1] Prediction Type : classification
```

```
## [1] Number of Trees (ntree) : 500
```

```
## [1] mtry value : 3
```

```
## [1] ====================================
```

```
## [1] ================================
```

```
## [1] The Table of predictions is :
```

```
##
##   <30   >30    NO
##     0  2545 13739
```

```
## [1] ================================
```

```
## [1] The Table of actual values is :
```

```
##
##  <30  >30   NO
## 1817 5688 8779
```

```
## [1] ================================
```

```
## [1] The Overall Accuracy is :
```

```
## [1] 0.5794645
```

```
## [1] ================================
```

```
## [1] The confusion matrix is :
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  <30  >30   NO
##        <30    0    0    0
##        >30  484 1359  702
##        NO  1333 4329 8077
##
## Overall Statistics
##
##                Accuracy : 0.5795
##                  95% CI : (0.5718, 0.5871)
##     No Information Rate : 0.5391
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1427
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: <30 Class: >30 Class: NO
## Sensitivity              0.0000    0.23892    0.9200
## Specificity              1.0000    0.88807    0.2456
## Pos Pred Value              NaN    0.53399    0.5879
```

```
## Neg Pred Value           0.8884    0.68491    0.7242
## Prevalence               0.1116    0.34930    0.5391
## Detection Rate           0.0000    0.08346    0.4960
## Detection Prevalence     0.0000    0.15629    0.8437
## Balanced Accuracy        0.5000    0.56350    0.5828

## [1] ================================

## [1] The first 10 Features in order of decreasing importance in prediction are:

## [1] ============================================
```

Table 16: MIC Prediction - Variable Importance (MeanDecreaseGini)

Col Id	Importance	Names
1	1319.1149	patient_nbr
17	981.2028	number_inpatient
18	963.1743	diag_1
12	952.9306	num_lab_procedures
19	938.7680	diag_2
20	917.6653	diag_3
14	910.9416	num_medications
9	673.2872	time_in_hospital
7	630.4113	discharge_disposition_id
4	598.9446	age

```
## [1] ============================================
```

We can see that Random Forest is really struggling to Predict the "<30" class. Very strangely, it is not able to make a single prediction from the "<30" class.

4.2.4 Algorithms that work with only numerical inputs

Unlike the MIC dataset where the categorical variables were mostly of the ordinal type, in the d130 dataset, most of the categorical variables are just simple unordered factors.

To explore Algorithms like ANN and XGBoost, one-hot encoding of the categorical variables is necessary. Otherwise we will end up misconfiguring these algorithms and they will generate inaccurate predictions.

We will not explore these algorithms at this point because with the Current Number of Unique Values for "patient_nbr" (71518), "diag_1" (717), "diag_2" (749) and "diag_3" (790), any kind of one-hot encoding is far too unwieldy and causes memory issues on stock computing hardware.

We will explore them at the end of the next section.

There are two new ML Algorithms that we have not used so far, ADABoost and ADABag which are quite popular within the ML community. Please see note about them after the Analysis for XGBoost.

4.3 Dataset Specification for CVD and Initial Analysis

Let us now consider sub-setting our d130 dataset to suit our initial premise of evaluating datasets related to Cardiovascular Diseases.

Let us consult our modern encyclopaedia of knowledge to see what the ICD9 is

https://en.wikipedia.org/wiki/International_Classification_of_Diseases

and what the ICD9 Codes mean

https://en.wikipedia.org/wiki/List_of_ICD-9_codes

Look up the diabetes and related classification codes

https://en.wikipedia.org/wiki/List_of_ICD-9_codes_240%E2%80%93279:_endocrine,_nutritional_and_metabolic_diseases,_and_immunity_disorders

Look up the CVD and related classification codes

https://en.wikipedia.org/wiki/List_of_ICD-9_codes_390%E2%80%93459:_diseases_of_the_circulatory_system

We will record the following for the entire d130 dataset

1. Number of unique patient_nbr
2. Number of unique "diag_1" codes
3. Number of unique "diag_2" codes
4. Number of unique "diag_3" codes

```
## [1] ============================
```

```
## [1] The Total Number of Unique Patients in the entire d130 dataset is :
## [2] 71518
```

```
## [1] The Total Number of Unique 'diag_1' codes in the entire d130 dataset is :
## [2] 717
```

```
## [1] The Total Number of Unique 'diag_2' codes in the entire d130 dataset is :
## [2] 749
```

```
## [1] The Total Number of Unique 'diag_3' codes in the entire d130 dataset is :
## [2] 790
```

```
## [1] ============================
```

We will create some simple regular expressions that can be used to filter our data according to the ICD9 codes for CVD and diabetes.

We will record

1. The list of unique "diag_1" codes related to CVD
2. The number of unique "diag_1" codes related to CVD
3. The number of observations in 'diag_1' referring to CVD

4. The list of unique “diag_2” codes related to diabetes
5. The number of unique “diag_2” codes related to diabetes
6. The number of observations in ‘diag_3’ referring to diabetes

---

7. The list of unique “diag_3” codes related to diabetes
8. The number of unique “diag_3” codes related to diabetes
9. The number of observations in ‘diag_3’ referring to diabetes

This will give us a view of how many cases have CVD as the Primary diagnosis and how many have diabetes as the Secondary and Additional Secondary diagnosis.

```
## [1] ==============================

## [1] " The list of Unique 'diag_1' codes referring to CVD is :"

##  [1] 414 428 398 410 402 427 411 423 426 404 416 424 415 429 413 420 785 397 395
## [20] 421 425 396 394 417 412 422

## [1] The Total Number of Unique 'diag_1' codes referring to CVD is :
## [2] 26

## [1] The Total Number of Observations in 'diag_1' referring to CVD is :
## [2] 22478

## [1] ==============================

## [1] ==============================

## [1] " The list of Unique 'diag_2' codes referring to Diabetes is :"

##  [1] 250.01 250    250.43 250.02 250.03 250.52 250.6  250.82 250.51 250.42
## [11] 250.41 250.13 250.12 250.53 250.93 250.7  250.83 250.11 250.81 250.5
## [21] 250.91 250.92 250.4  250.23 250.22 250.8  250.9  250.1  250.2  250.31
## [31] 250.33 250.32 250.21 250.3

## [1] The Total Number of Unique 'diag_2' codes referring to Diabetes is :
## [2] 34

## [1] The Total Number of Observations in 'diag_2' referring to Diabetes is :
## [2] 12794

## [1] ==============================

## [1] ==============================
```

```
## [1] " The list of Unique 'diag_3' codes referring to Diabetes is :"

##  [1] 250    250.6  250.01 250.42 250.41 250.02 250.8  250.7  250.52 250.82
## [11] 250.03 250.4  250.51 250.43 250.92 250.23 250.53 250.83 250.5  250.91
## [21] 250.13 250.12 250.81 250.9  250.22 250.93 250.2  250.1  250.11 250.3
## [31] 250.21 250.31

## [1] The Total Number of Unique 'diag_3' codes referring to Diabetes is :
## [2] 32

## [1] The Total Number of Observations in 'diag_3' referring to Diabetes is :
## [2] 17157

## [1] ============================
```

We will remove observations that are not relevant for our analysis. These are observations where the patient expired during treatment, was moved to hospice or discharged to a long term care hospital.

To keep our ANN manageable and be able to run them on stock computing hardware without taking too much time, we will filter our data where the "diag_1" column contains ICD9 codes related to CVD and the "diag_2" or "diag_3" columns contain ICD9 codes related to diabetes.

We will collect and record some details regarding our newly created subset.

```
## [1] The Number of Unique Patients for the Dataset filtered for CVD cases with diabetes is :
## [2] 5500

## [1] ============================

## [1] "The Structure of the Filtered Datset is :"

## 'data.frame':    5910 obs. of  50 variables:
##  $ encounter_id            : int  35754 63768 40926 325848 383430 486156 682494 961464 1070256 10725
##  $ patient_nbr             : int  82637451 114882984 85504905 63023292 80588529 86240259 64729746 10
##  $ race                    : chr  "Caucasian" "Caucasian" "Caucasian" "Caucasian" ...
##  $ gender                  : chr  "Male" "Male" "Female" "Female" ...
##  $ age                     : chr  "[50-60)" "[70-80)" "[40-50)" "[60-70)" ...
##  $ weight                  : chr  "?" "?" "?" "?" ...
##  $ admission_type_id       : int  2 1 1 1 1 3 1 1 2 3 ...
##  $ discharge_disposition_id: int  1 1 3 1 2 5 3 1 1 1 ...
##  $ admission_source_id     : int  2 7 7 7 7 4 7 7 4 2 ...
##  $ time_in_hospital        : int  3 5 7 2 1 9 3 4 3 3 ...
##  $ payer_code              : chr  "?" "?" "?" "?" ...
##  $ medical_specialty       : chr  "?" "?" "Family/GeneralPractice" "Cardiology" ...
##  $ num_lab_procedures      : int  31 73 60 41 28 25 64 51 41 29 ...
##  $ num_procedures          : int  6 0 0 0 0 3 0 0 4 1 ...
##  $ num_medications         : int  16 12 15 11 15 16 15 9 20 7 ...
##  $ number_outpatient       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ number_emergency        : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ number_inpatient        : int  0 0 0 0 0 2 1 0 1 0 ...
##  $ diag_1                  : chr  "414" "428" "428" "411" ...
##  $ diag_2                  : chr  "411" "492" "250.43" "250.01" ...
##  $ diag_3                  : chr  "250" "250" "250.6" "401" ...
```

```
##  $ number_diagnoses        : int  9 8 8 6 4 7 9 4 7 9 ...
##  $ max_glu_serum           : chr  "None" "None" "None" "None" ...
##  $ A1Cresult               : chr  "None" "None" "None" "None" ...
##  $ metformin               : chr  "No" "No" "Steady" "No" ...
##  $ repaglinide             : chr  "No" "No" "Up" "No" ...
##  $ nateglinide             : chr  "No" "No" "No" "No" ...
##  $ chlorpropamide          : chr  "No" "No" "No" "No" ...
##  $ glimepiride             : chr  "No" "No" "No" "No" ...
##  $ acetohexamide           : chr  "No" "No" "No" "No" ...
##  $ glipizide               : chr  "No" "No" "No" "No" ...
##  $ glyburide               : chr  "No" "Steady" "No" "No" ...
##  $ tolbutamide             : chr  "No" "No" "No" "No" ...
##  $ pioglitazone            : chr  "No" "No" "No" "No" ...
##  $ rosiglitazone           : chr  "No" "No" "No" "Steady" ...
##  $ acarbose                : chr  "No" "No" "No" "No" ...
##  $ miglitol                : chr  "No" "No" "No" "No" ...
##  $ troglitazone            : chr  "No" "No" "No" "No" ...
##  $ tolazamide              : chr  "No" "No" "No" "No" ...
##  $ examide                 : chr  "No" "No" "No" "No" ...
##  $ citoglipton             : chr  "No" "No" "No" "No" ...
##  $ insulin                 : chr  "Steady" "No" "Down" "Down" ...
##  $ glyburide.metformin     : chr  "No" "No" "No" "No" ...
##  $ glipizide.metformin     : chr  "No" "No" "No" "No" ...
##  $ glimepiride.pioglitazone: chr  "No" "No" "No" "No" ...
##  $ metformin.rosiglitazone : chr  "No" "No" "No" "No" ...
##  $ metformin.pioglitazone  : chr  "No" "No" "No" "No" ...
##  $ change                  : chr  "No" "No" "Ch" "Ch" ...
##  $ diabetesMed             : chr  "Yes" "Yes" "Yes" "Yes" ...
##  $ readmitted              : chr  ">30" ">30" "<30" ">30" ...

## [1] ============================
```

We will modify our newly created dataset with the right variable types. We will then partition the dataset into Training and Testing sets and the Training set further into Training CV and Testing CV datasets.

```
## [1] ============================

## [1] "The Structure of the Filtered Datset with the right variable types is :"

## 'data.frame':    5910 obs. of  50 variables:
##  $ encounter_id            : int  35754 63768 40926 325848 383430 486156 682494 961464 1070256 107255
##  $ patient_nbr             : Factor w/ 5500 levels "1314","31320",..: 4152 5339 4303 3705 4106 4358 3
##  $ race                    : Factor w/ 5 levels "AfricanAmerican",..: 3 3 3 3 3 3 1 3 3 3 ...
##  $ gender                  : Factor w/ 2 levels "Female","Male": 2 2 1 1 1 1 1 1 1 2 ...
##  $ age                     : Factor w/ 9 levels "[10-20)","[20-30)",..: 5 7 4 6 7 7 4 7 5 7 ...
##  $ weight                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ admission_type_id       : Factor w/ 6 levels "1","2","3","5",..: 2 1 1 1 1 3 1 1 2 3 ...
##  $ discharge_disposition_id: Factor w/ 19 levels "1","2","3","4",..: 1 1 3 1 2 5 3 1 1 1 ...
##  $ admission_source_id     : Factor w/ 11 levels "1","2","3","4",..: 2 7 7 7 7 4 7 7 4 2 ...
##  $ time_in_hospital        : int  3 5 7 2 1 9 3 4 3 3 ...
##  $ payer_code              : Factor w/ 15 levels "BC","CH","CM",..: NA NA NA NA NA NA NA NA NA NA ..
##  $ medical_specialty       : Factor w/ 26 levels "AllergyandImmunology",..: NA NA 5 2 2 10 NA 2 NA NA
##  $ num_lab_procedures      : int  31 73 60 41 28 25 64 51 41 29 ...
```

```
##  $ num_procedures          : int  6 0 0 0 0 3 0 0 4 1 ...
##  $ num_medications         : int  16 12 15 11 15 16 15 9 20 7 ...
##  $ number_outpatient       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ number_emergency        : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ number_inpatient        : int  0 0 0 0 0 2 1 0 1 0 ...
##  $ diag_1                  : Factor w/ 22 levels "395","396","398",..: 9 20 20 7 9 20 20 9 9 19 ...
##  $ diag_2                  : Factor w/ 211 levels "135","151","153",..: 95 129 30 18 95 105 20 95 95
##  $ diag_3                  : Factor w/ 186 levels "112","138","174",..: 10 10 28 76 11 11 148 10 10
##  $ number_diagnoses        : int  9 8 8 6 4 7 9 4 7 9 ...
##  $ max_glu_serum           : Factor w/ 4 levels ">200",">300",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ A1Cresult               : Factor w/ 4 levels ">7",">8","None",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ metformin               : Factor w/ 4 levels "Down","No","Steady",..: 2 2 3 2 3 2 2 2 2 3 ...
##  $ repaglinide             : Factor w/ 4 levels "Down","No","Steady",..: 2 2 4 2 2 2 2 2 2 2 ...
##  $ nateglinide             : Factor w/ 3 levels "No","Steady",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ chlorpropamide          : Factor w/ 4 levels "Down","No","Steady",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ glimepiride             : Factor w/ 4 levels "Down","No","Steady",..: 2 2 2 2 2 2 2 3 2 2 ...
##  $ acetohexamide           : Factor w/ 1 level "No": 1 1 1 1 1 1 1 1 1 1 ...
##  $ glipizide               : Factor w/ 4 levels "Down","No","Steady",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ glyburide               : Factor w/ 4 levels "Down","No","Steady",..: 2 3 2 2 2 3 2 2 2 2 ...
##  $ tolbutamide             : Factor w/ 2 levels "No","Steady": 1 1 1 1 1 1 1 1 1 1 ...
##  $ pioglitazone            : Factor w/ 4 levels "Down","No","Steady",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ rosiglitazone           : Factor w/ 4 levels "Down","No","Steady",..: 2 2 2 3 2 2 2 2 2 2 ...
##  $ acarbose                : Factor w/ 3 levels "No","Steady",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ miglitol                : Factor w/ 2 levels "No","Steady": 1 1 1 1 1 1 1 1 1 1 ...
##  $ troglitazone            : Factor w/ 2 levels "No","Steady": 1 1 1 1 1 1 1 1 1 1 ...
##  $ tolazamide              : Factor w/ 2 levels "No","Steady": 1 1 1 1 1 1 1 1 1 1 ...
##  $ examide                 : Factor w/ 1 level "No": 1 1 1 1 1 1 1 1 1 1 ...
##  $ citoglipton             : Factor w/ 1 level "No": 1 1 1 1 1 1 1 1 1 1 ...
##  $ insulin                 : Factor w/ 4 levels "Down","No","Steady",..: 3 2 1 1 1 1 4 2 3 3 ...
##  $ glyburide.metformin     : Factor w/ 3 levels "Down","No","Steady": 2 2 2 2 2 2 2 2 2 2 ...
##  $ glipizide.metformin     : Factor w/ 1 level "No": 1 1 1 1 1 1 1 1 1 1 ...
##  $ glimepiride.pioglitazone: Factor w/ 1 level "No": 1 1 1 1 1 1 1 1 1 1 ...
##  $ metformin.rosiglitazone : Factor w/ 1 level "No": 1 1 1 1 1 1 1 1 1 1 ...
##  $ metformin.pioglitazone  : Factor w/ 1 level "No": 1 1 1 1 1 1 1 1 1 1 ...
##  $ change                  : Factor w/ 2 levels "Ch","No": 2 2 1 1 1 1 1 2 2 1 ...
##  $ diabetesMed             : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ readmitted              : Factor w/ 3 levels "<30",">30","NO": 2 2 1 2 2 1 2 2 2 3 ...

## [1] ==============================

## [1] ================================

## [1] The Table of Outcomes is

##
##  <30  >30   NO
##  620 2101 3189

## [1] The Ratio of Patients who were NOT readmitted is :
## [2] 0.5396

## [1] The Ratio of Patients who were readmitted in >30 days is :
## [2] 0.3555
```

```
## [1] The Ratio of Patients who were readmitted in <30 days is :
## [2] 0.1049
```

4.3.1 Initial Analysis

4.3.1.1 Naive Bayes As earlier, let us start with Naive Bayes to get some sense of where we can get with our predictions.

4.3.1.1.1 Categorical Classification We will use Poisson for all discrete variables, Gaussian for weight (which is not relevant anyway) and Laplace smoothing with a value of 1.

```
## [1] The Naive Bayes Summary is :

##
## ================================= Naive Bayes ==================================
##
## - Call: naive_bayes.default(x = d130_cvd_modified_cv_train_set[, 2:49],      y = d130_cvd_modified_cv
## - Laplace: 1
## - Classes: 3
## - Samples: 3780
## - Features: 48
## - Conditional distributions:
##     - Bernoulli: 7
##     - Categorical: 32
##     - Poisson: 8
##     - Gaussian: 1
## - Prior probabilities:
##     - <30: 0.1048
##     - >30: 0.3556
##     - NO: 0.5397
##
## --------------------------------------------------------------------------------

## [1] ================================

## [1] The Table of predictions is :

##
## <30 >30  NO
##  68 312 567

## [1] ================================

## [1] The Table of actual values is :

##
## <30 >30  NO
## 100 336 511

## [1] ================================

## [1] The Overall Accuracy is :

## [1] 0.5575502
```

```
## [1] ================================
```

```
## [1] The confusion matrix is :
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <30 >30  NO
##        <30  10  27  31
##        >30  46 152 114
##        NO   44 157 366
##
## Overall Statistics
##
##                Accuracy : 0.5576
##                  95% CI : (0.5253, 0.5895)
##     No Information Rate : 0.5396
##     P-Value [Acc > NIR] : 0.140989
##
##                   Kappa : 0.1991
##
##  Mcnemar's Test P-Value : 0.002876
##
## Statistics by Class:
##
##                      Class: <30 Class: >30 Class: NO
## Sensitivity             0.10000     0.4524    0.7162
## Specificity             0.93152     0.7381    0.5390
## Pos Pred Value          0.14706     0.4872    0.6455
## Neg Pred Value          0.89761     0.7102    0.6184
## Prevalence              0.10560     0.3548    0.5396
## Detection Rate          0.01056     0.1605    0.3865
## Detection Prevalence    0.07181     0.3295    0.5987
## Balanced Accuracy       0.51576     0.5953    0.6276
```

```
## [1] ================================
```

4.3.1.1.2 Binary Classification Let us see if we can do better with a Binary Classifier, As earlier, for the "readmitted" field, we will retain the "NO" value as-is. We will consolidate the ">30" and "<30" values into a much simpler "YES".

```
## [1] The Naive Bayes Summary is :

## [1] ================================

##
## ================================ Naive Bayes ================================
##
## - Call: naive_bayes.default(x = d130_cvd_modified_cv_train_set_binary[,      2:49], y = d130_cvd_mod
## - Laplace: 1
## - Classes: 2
## - Samples: 947
## - Features: 48
## - Conditional distributions:
##     - Bernoulli: 7
##     - Categorical: 32
##     - Poisson: 8
##     - Gaussian: 1
## - Prior probabilities:
##     - NO: 0.5396
##     - YES: 0.4604
##
## ----------------------------------------------------------------------------

## [1] ================================

## [1] ================================

## [1] The Table of predictions is :

##
##  NO YES
## 533 414

## [1] ================================

## [1] The Table of actual values is :

##
##  NO YES
## 511 436

## [1] ================================

## [1] The Overall Accuracy is :

## [1] 0.7782471
```

```
## [1] ================================
```

```
## [1] The confusion matrix is :
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NO YES
##        NO  417 116
##        YES  94 320
##
##                Accuracy : 0.7782
##                  95% CI : (0.7504, 0.8043)
##     No Information Rate : 0.5396
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.552
##
##  Mcnemar's Test P-Value : 0.1473
##
##             Sensitivity : 0.8160
##             Specificity : 0.7339
##          Pos Pred Value : 0.7824
##          Neg Pred Value : 0.7729
##              Prevalence : 0.5396
##          Detection Rate : 0.4403
##    Detection Prevalence : 0.5628
##       Balanced Accuracy : 0.7750
##
##        'Positive' Class : NO
##
```

```
## [1] ================================
```

We have a dramatic improvement in the overall accuracy as well as the balanced accuracy.

As in the previous section with MIC, our predictions using Naive Bayes using a binary classifier are much better.

This is one of the ways in which we can increase the accuracy of our predictions provided we have the flexibility to modify the outcome.

4.3.1.2 Random Forest Let us check what Random Forest can do for the CVD dataset. Our hopes are tempered by what we have seen already.

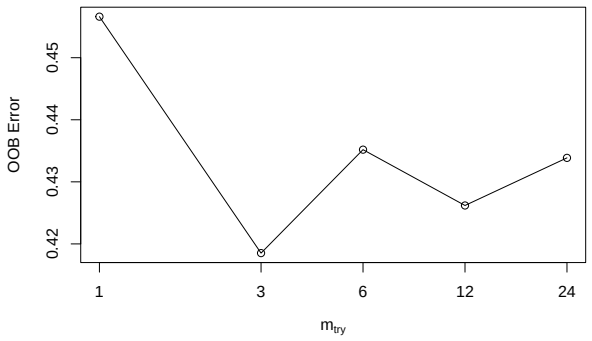
As earlier, we will

1. Zero out the columns related to weight
2. We will retain the "?" to represent missing values in payer_code, medical_specialty and other columns

```
## [1] ====================================

## [1] The Random Forest mtry values and error rates are

## mtry = 6  OOB error = 43.52%
## Searching left ...
## mtry = 12    OOB error = 42.62%
## 0.02066869 1e-05
## mtry = 24    OOB error = 43.39%
## -0.01800124 1e-05
## Searching right ...
## mtry = 3     OOB error = 41.85%
## 0.01800124 1e-05
## mtry = 1     OOB error = 45.66%
## -0.09102402 1e-05
```




```
## [1] ====================================

## [1]  Details for Random Forest  for the d130 Dataset are:

## [1] Prediction Type : classification

## [1] Number of Trees (ntree) : 500

## [1] mtry value : 3

## [1] ====================================

## [1] ================================
```

```
## [1] The Table of predictions is :

##
## <30 >30  NO
##   0  60 887

## [1] ================================

## [1] The Table of actual values is :

##
## <30 >30  NO
## 100 336 511

## [1] ================================

## [1] The Overall Accuracy is :

## [1] 0.5564942

## [1] ================================

## [1] The confusion matrix is :

## Confusion Matrix and Statistics
##
##           Reference
## Prediction <30 >30  NO
##        <30   0   0   0
##        >30  12  32  16
##        NO   88 304 495
##
## Overall Statistics
##
##                Accuracy : 0.5565
##                  95% CI : (0.5242, 0.5884)
##     No Information Rate : 0.5396
##     P-Value [Acc > NIR] : 0.1561
##
##                   Kappa : 0.0606
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: <30 Class: >30 Class: NO
## Sensitivity              0.0000    0.09524    0.9687
## Specificity              1.0000    0.95417    0.1009
## Pos Pred Value              NaN    0.53333    0.5581
## Neg Pred Value           0.8944    0.65727    0.7333
## Prevalence               0.1056    0.35480    0.5396
## Detection Rate           0.0000    0.03379    0.5227
## Detection Prevalence     0.0000    0.06336    0.9366
## Balanced Accuracy        0.5000    0.52471    0.5348
```

```
## [1] The first 10 Features in order of decreasing importance in prediction are:

## [1] ============================================
```

Table 17: MIC Prediction - Variable Importance (MeanDecreaseGini)

Col Id	Importance	Names
1	101.09112	patient_nbr
12	74.49560	num_lab_procedures
14	72.79874	num_medications
19	66.23363	diag_2
18	56.57905	diag_1
20	55.51932	diag_3
21	55.05278	number_diagnoses
17	53.65783	number_inpatient
9	50.77263	time_in_hospital
13	47.54967	num_procedures

```
## [1] ============================================
```

As earlier, Random Forest is struggling badly with the CVD dataset too.

The list of important features has changed a bit with some features being reordered and some being replaced by others. The most important feature patient_nbr remains unchanged.

As earlier, Random Forest is unable to make even a single prediction in the "<30" class. We will no longer explore Random Forest for this dataset.

4.3.1.3 XGBoost

Let us see what kind of results we can obtain for the CVD dataset with XGBoost.

Before that we need to modify our dataset and expand the categorical variables using one-hot encoding. We will modify the dataset so that it is suitable for processing by XGBoost directly.

```
## [1] ================================

## [1] The XGBoost Summary is :

## ##### xgb.Booster
## call:
##   xgb.train(params = list(num_class = 8, booster = "gbtree"), data = dtrain_d130_cvd,
##     nrounds = 240, objective = "multi:softmax", verbose = 0,
##     max_depth = 10, eta = 0.15, nthread = 8)
## # of features: 6086
## # of rounds:  240

## [1] ================================

## [1] ================================

## [1] The Table of predictions is :

##
## <30 >30  NO
##   8 321 618

## [1] The Table of actual values is :

##
## <30 >30  NO
## 100 336 511

## [1] ================================

## [1] The Overall Accuracy is :

## [1] 0.5702218

## [1] ================================

## [1] The confusion matrix is :

## Confusion Matrix and Statistics
##
##           Reference
## Prediction <30 >30  NO
##        <30   2   4   2
##        >30  46 152 123
##        NO   52 180 386
##
```

```
## Overall Statistics
##
##                Accuracy : 0.5702
##                  95% CI : (0.538, 0.602)
##     No Information Rate : 0.5396
##     P-Value [Acc > NIR] : 0.0314
##
##                   Kappa : 0.184
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: <30 Class: >30 Class: NO
## Sensitivity            0.020000     0.4524    0.7554
## Specificity            0.992916     0.7234    0.4679
## Pos Pred Value         0.250000     0.4735    0.6246
## Neg Pred Value         0.895634     0.7061    0.6201
## Prevalence             0.105597     0.3548    0.5396
## Detection Rate         0.002112     0.1605    0.4076
## Detection Prevalence   0.008448     0.3390    0.6526
## Balanced Accuracy      0.506458     0.5879    0.6116

## [1] ================================
```

XGBoost is able to predict some observations in the "<30" class but just two are correctly categorised. It is doing much better for the ">30" and "NO" classes. However, we are far from having something useful from XGBoost for either the shape of the classes or the predictions for individual observations within the "<30" and ">30" classes.

4.3.1.4 ADABoost and ADABag I have not included ADABoost and ADABag in the list of algorithms as they are unable to make any predictions in the "<30" class with some reasonable parameters. Even with these parameters ADABoost and ADABag take a very long time to run compared to XGBoost. With parallel processing turned on, I have see them take up 8 cores (16 Threads with Hyperthreading) and almost 40GB of memory which is way above what XGBoost uses.

4.3.1.5 Artificial Neural Networks Let us try ANN Next, We have a much smaller dataset to work with so hopefully, we should be able to load the CVD dataset into Memory and process it.

4.3.1.5.1 Dataset Preparation The dataset is by very nature complex and the expansion of the Categorical variables using "one-hot" encoding makes it even more unwieldy. We have to look at how best we can model the same using ANN.

We will first try to classify our predictors/features into related groups.

We will remove factors with a single level as they are not useful in prediction and also cause errors in one-hot encoding.

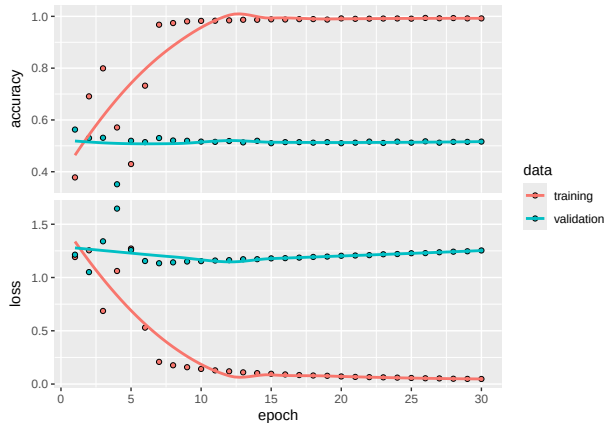
For one-hot encoding of Categorical Variables, we will use the "one_hot" function from the "mltools" package as it is a lot easier to use for our current case.

Once we have performed the one-hot encoding, for the Functional API, we will collect the column indices of the predictors/features as per the respective groups. This will ensure that our Functional API has all the variables which are one-hot encoded assigned to the correct groups. The code is not very complex or difficult to follow but has subtleties that we need to be aware of, particularly around variable identification using strings.We have already seen this with our MIC dataset.

For both Functional and Sequential API alike, we will need to scale the variables and centre the Testing set around the Training set.

4.3.1.5.2 Functional API Let us check what ANN built with Functional API can do

```
## [1] "Plot of metrics and trends during training"
```




```
## 30/30 - 0s - 10ms/step
```

```
## [1] ================================
```

```
## [1] The Table of predictions is :
```

```
##
## <30 >30  NO
## 100 354 493
```

```
## [1] ================================
```

```
## [1] The Table of actual values is :

##
## <30 >30  NO
## 100 336 511

## [1] ================================

## [1] The Overall Accuracy is :

## [1] 0.5163675

## [1] ================================

## [1] The Confusion Matrix is :

## Confusion Matrix and Statistics
##
##           Reference
## Prediction <30 >30  NO
##        <30  14  45  41
##        >30  43 158 153
##        NO   43 133 317
##
## Overall Statistics
##
##                Accuracy : 0.5164
##                  95% CI : (0.484, 0.5486)
##     No Information Rate : 0.5396
##     P-Value [Acc > NIR] : 0.9287
##
##                   Kappa : 0.1594
##
##  Mcnemar's Test P-Value : 0.6842
##
## Statistics by Class:
##
##                      Class: <30 Class: >30 Class: NO
## Sensitivity             0.14000     0.4702    0.6204
## Specificity             0.89847     0.6792    0.5963
## Pos Pred Value          0.14000     0.4463    0.6430
## Neg Pred Value          0.89847     0.6998    0.5727
## Prevalence              0.10560     0.3548    0.5396
## Detection Rate          0.01478     0.1668    0.3347
## Detection Prevalence    0.10560     0.3738    0.5206
## Balanced Accuracy       0.51923     0.5747    0.6083

## [1] ================================
```

We have excellent shape to our predictions. We are even able to improve upon the predictions made for the "<30" and ">30" classes. The overall accuracy is poorer than XGBoost because XGBoost can predict live cases far better.

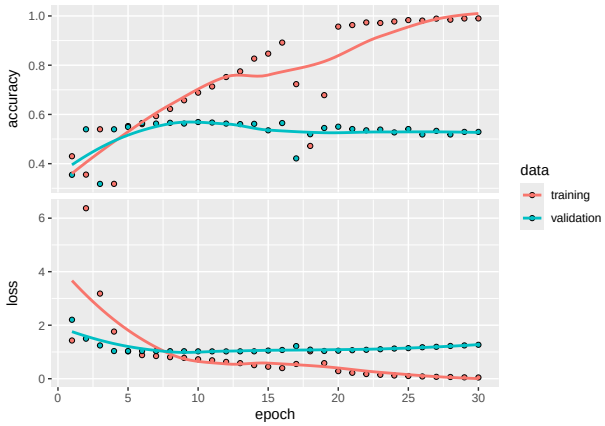
4.3.1.5.3 Sequential API

We now turn our attention to ANN built with Sequential API

```
## [1] 0
```

```
## [1] 0
```

```
## [1] "Plot of metrics and trends during training"
```




```
## 30/30 - 1s - 23ms/step
```

```
## [1] ================================
```

```
## [1] The Table of predictions is :
```

```
##
## <30 >30  NO
##  83 335 529
```

```
## [1] ================================
```

```
## [1] The Table of actual values is :
```

```
##
## <30 >30  NO
## 100 336 511
```

```
## [1] ================================
```

```
## [1] The Overall Accuracy is :
```

```
## [1] 0.5290391
```

```
## [1] ================================
```

```
## [1] The Confusion Matrix is :
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <30 >30  NO
##        <30  12  36  35
##        >30  42 153 140
##        NO   46 147 336
##
## Overall Statistics
##
##                Accuracy : 0.529
##                  95% CI : (0.4967, 0.5612)
##     No Information Rate : 0.5396
##     P-Value [Acc > NIR] : 0.7533
##
##                   Kappa : 0.1647
##
##  Mcnemar's Test P-Value : 0.5467
##
## Statistics by Class:
##
##                      Class: <30 Class: >30 Class: NO
## Sensitivity             0.12000     0.4554    0.6575
## Specificity             0.91617     0.7021    0.5573
## Pos Pred Value          0.14458     0.4567    0.6352
## Neg Pred Value          0.89815     0.7010    0.5813
## Prevalence              0.10560     0.3548    0.5396
## Detection Rate          0.01267     0.1616    0.3548
## Detection Prevalence    0.08765     0.3537    0.5586
## Balanced Accuracy       0.51809     0.5787    0.6074

## [1] ================================
```

With Sequential API, we do not have as much success as Functional API in increasing the Predictions for the "<30" and ">30" classes. Instead we have an improvement in predictions for the "NO" class.

Again not to belabour the point. It all depends on how we tune our ANN

4.3.2 Predictions for Holdout Set

Let us evaluate the performance of our Algorithms against the Holdout set

4.3.2.1 Naive Bayes

We evaluate Naive Bayes first

```
## [1] ================================
```

```
## [1] The Table of predictions is :
```

```
##
## <30 >30  NO
##  96 457 630
```

```
## [1] ================================
```

```
## [1] The Table of actual values is :
```

```
##
## <30 >30  NO
## 124 421 638
```

```
## [1] ================================
```

```
## [1] The Overall Accuracy is :
```

```
## [1] 0.5621302
```

```
## [1] ================================
```

```
## [1] The confusion matrix is :
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <30 >30  NO
##        <30  14  38  44
##        >30  66 224 167
##        NO   44 159 427
##
## Overall Statistics
##
##                Accuracy : 0.5621
##                  95% CI : (0.5333, 0.5906)
##     No Information Rate : 0.5393
##     P-Value [Acc > NIR] : 0.06094
##
##                   Kappa : 0.2275
##
##  Mcnemar's Test P-Value : 0.05182
```

```
##
## Statistics by Class:
##
##                      Class: <30 Class: >30 Class: NO
## Sensitivity             0.11290     0.5321    0.6693
## Specificity             0.92257     0.6942    0.6275
## Pos Pred Value          0.14583     0.4902    0.6778
## Neg Pred Value          0.89880     0.7287    0.6184
## Prevalence              0.10482     0.3559    0.5393
## Detection Rate          0.01183     0.1893    0.3609
## Detection Prevalence    0.08115     0.3863    0.5325
## Balanced Accuracy       0.51774     0.6131    0.6484

## [1] ================================
```

The shape of predictions with Naive Bayes is quite decent. The Predictions within the "<30" class are quite poor. Predictions within the ">30" class have improved.

4.3.2.2 XGBoost Let us evaluate XGBoost

```
## [1] ================================

## [1] The Table of predictions is :

##
## <30 >30  NO
##  21 406 756

## [1] ================================

## [1] The Table of actual values is :

##
## <30 >30  NO
## 124 421 638

## [1] ================================

## [1] The Overall Accuracy is :

## [1] 0.5857988

## [1] ================================

## [1] The confusion matrix is :

## Confusion Matrix and Statistics
##
##           Reference
## Prediction <30 >30  NO
##        <30   5  10   6
##        >30  54 204 148
##        NO   65 207 484
##
## Overall Statistics
##
##                Accuracy : 0.5858
##                  95% CI : (0.5571, 0.614)
##     No Information Rate : 0.5393
##     P-Value [Acc > NIR] : 0.0007163
##
##                   Kappa : 0.2205
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: <30 Class: >30 Class: NO
## Sensitivity            0.040323     0.4846    0.7586
```

```
## Specificity            0.984891     0.7349    0.5009
## Pos Pred Value         0.238095     0.5025    0.6402
## Neg Pred Value         0.897590     0.7207    0.6393
## Prevalence             0.104818     0.3559    0.5393
## Detection Rate         0.004227     0.1724    0.4091
## Detection Prevalence   0.017751     0.3432    0.6391
## Balanced Accuracy      0.512607     0.6097    0.6298
```

```
## [1] ================================
```

XGBoost continues to struggle with the shape of the predictions and predictions with the "<30" class. It achieves greater accuracy by making more predictions in the ">30" and "NO" classes.

XGBoost too has plenty of tuning parameters that we can explore to better match the shape and increase accuracy.

4.3.2.3 Artificial Neural Networks

4.3.2.3.1 Functional API Next we evaluate ANN built using Functional API

```
## 37/37 - 0s - 9ms/step
```

```
## [1] ================================
```

```
## [1] The Table of predictions is :
```

```
##
## <30 >30  NO
## 133 441 609
```

```
## [1] ================================
```

```
## [1] The Table of actual values is :
```

```
##
## <30 >30  NO
## 124 421 638
```

```
## [1] ================================
```

```
## [1] The Overall Accuracy is :
```

```
## [1] 0.5316991
```

```
## [1] ================================
```

```
## [1] The confusion matrix is :
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <30 >30  NO
##        <30  21  56  56
##        >30  57 205 179
##        NO   46 160 403
##
## Overall Statistics
##
##                Accuracy : 0.5317
##                  95% CI : (0.5028, 0.5604)
##     No Information Rate : 0.5393
##     P-Value [Acc > NIR] : 0.7104
##
##                   Kappa : 0.1897
##
##  Mcnemar's Test P-Value : 0.5612
```

```
##
## Statistics by Class:
##
##                      Class: <30 Class: >30 Class: NO
## Sensitivity             0.16935     0.4869    0.6317
## Specificity             0.89424     0.6903    0.6220
## Pos Pred Value          0.15789     0.4649    0.6617
## Neg Pred Value          0.90190     0.7089    0.5906
## Prevalence              0.10482     0.3559    0.5393
## Detection Rate          0.01775     0.1733    0.3407
## Detection Prevalence    0.11243     0.3728    0.5148
## Balanced Accuracy       0.53180     0.5886    0.6268

## [1] ================================
```

ANN Built using the Functional API offer the best shape as yet. They also offer the most number of accurate predictions within the "<30" class.

4.3.2.3.2 Sequential API

Finally we evaluate ANN built using Sequential API

```
## [1] 0
```

```
## [1] 0
```

```
## 37/37 - 0s - 5ms/step
```

```
## [1] ================================
```

```
## [1] The Table of predictions is :
```

```
##
## <30 >30  NO
## 143 455 585
```

```
## [1] ================================
```

```
## [1] The Table of actual values is :
```

```
##
## <30 >30  NO
## 124 421 638
```

```
## [1] ================================
```

```
## [1] The Overall Accuracy is :
```

```
## [1] 0.5384615
```

```
## [1] ================================
```

```
## [1] The Confusion Matrix is :
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <30 >30  NO
##        <30  17  66  60
##        >30  61 218 176
##        NO   46 137 402
##
## Overall Statistics
##
##                Accuracy : 0.5385
##                  95% CI : (0.5096, 0.5672)
##     No Information Rate : 0.5393
##     P-Value [Acc > NIR] : 0.53516
##
##                   Kappa : 0.2094
```

```
##
##  Mcnemar's Test P-Value : 0.07498
##
## Statistics by Class:
##
##                      Class: <30 Class: >30 Class: NO
## Sensitivity             0.13710     0.5178    0.6301
## Specificity             0.88102     0.6890    0.6642
## Pos Pred Value          0.11888     0.4791    0.6872
## Neg Pred Value          0.89712     0.7212    0.6054
## Prevalence              0.10482     0.3559    0.5393
## Detection Rate          0.01437     0.1843    0.3398
## Detection Prevalence    0.12088     0.3846    0.4945
## Balanced Accuracy       0.50906     0.6034    0.6472
```

Sequential API offer decent shape, and slightly better predictions in the "<30" class when compared to Naive Bayes. The predictions for the ">30" and "NO" classes is better than those provided by ANN built with functional API.

Let us compare and summarise the results for all of our models in the next section.

4.4 Results & Inference

The d130 dataset has lesser features than the MIC dataset but is far more difficult to make predictions against as all the algorithms struggle with the overall accuracy as well as the shape of predictions.

We have also seen that the predictions for the Validation set and the Holdout sets also vary between algorithms.

1. With ANN, depending on the way the ANN is built, the predictions can vary a lot. Another situation to watch out for with ANN is that when they are very well optimised for the Validation set, they often perform poorly against the Holdout set. The behaviour is similar to what we can see with over-training or over-fitting. This just as observed with the MIC dataset.
2. To reiterate what was stated in the MIC Analysis as well. XGBoost and ANN have a lot of tuning parameters that can greatly enhance their accuracy. Whatever I have built are just the most basic for purposes of comparison. With the right amount of skill and time, they can be tuned to offer far more accurate predictions.

Let us summarise and review our analysis so far

```
## [1] ================================

## [1] The Ratios of Outcomes for reference is

## [1] Ratio of NO : 0.0313

## [1] Ratio of >30 : 0.0206

## [1] Ratio of <30  0.0061

## [1] ================================
```

Table 18: d130 CVD - Results Summary

	Naive Bayes	XGBoost	ANN Sequential API	ANN Functional API
Accuracy	0.5621	0.5858	0.5385	0.5317

Table 19: d130 CVD - Table of distributions

readmitted	Actual Values	Naive Bayes	XGBoost	ANN Sequential API	ANN Functional API
<30	124	96	21	143	133
>30	421	457	406	455	441
NO	638	630	756	585	609

Table 20: d130 CVD - Table of Accurate Predictions

readmitted	Actual Values	Naive Bayes	XGBoost	ANN Sequential API	ANN Functional API
<30	124	14	5	17	21
>30	421	224	204	218	205
NO	638	427	484	402	403

In terms of Accuracy, XGBoost offers the best overall accuracy, but it achieves this by predicting more outcomes in the "NO" class which is counter productive to our goals for this dataset.

Naive Bayes provides perhaps the best results for the efforts expended. It has good accuracy comparatively and even the shape of the predictions is quite good. We must state here that the configuration efforts and computational resources required for Naive Bayes are very low compared to the other Algorithms.

Where all the Algorithms struggle is the ability to predict the outcomes within the "<30" class. The Sensitivity is extremely poor with all algorithms.

ANN built using Functional API provide the best balance in terms of overall accuracy and the shape of the predictions. The shape of the predictions is useful to those who are looking at a more higher level into the outcomes and the predictions at the level of the class and not individual observations within each class.

The efforts required are quite high and another issue with ANN is that they need to be tuned for every dataset against which we need to make predictions.

For this dataset, if quick results are desired, Naive Bayes is a much better choice.

For those who with the required skills and who are willing to spend the time and effort, ANN or even XGBoost can be a good choice.

5 Final Results

It has been a long project and fairly long report. With results summarised for each dataset in their respective sections.

We have seen various datasets and the performance of various algorithms

1. Heart Failure Predictions (HFP) dataset
    1. Extremely simple dataset with 12 features, no missing values and 5000 observations
    2. Simple binary outcome
    3. Random Forest is a clear choice both for accuracy and amount of effort required.
    4. ANN do a good job with results close to what Random Forest provides but require a lot more programming effort and computing resources.
2. Myocardial Infarction Complication (MIC) dataset
    1. Complex dataset with many features (111) and feature types (4)
    2. Much smaller number of observations (1700)
    3. Missing values in several features. Imputation possible
    4. Categorical outcome. Binary as an option.
    5. XGBoost is a very good choice for both accuracy and amount of effort required. Another big advantage with XGBoost is that it is extremely fast and requires a lot less computing resources when compared to ANN
    6. ANN are seemingly the better choice for those who have the skills, who can spend the time and effort. However XGBoost with adequate tuning can be just as good an option.
3. Diabetes 130-US Hospitals for Years 1999-2008 (d130) dataset
    1. Complex dataset with many features (48) and feature types (4)
    2. Larger number of observations (101766)
        1. We have subset the dataset to obtain 5910 observations for running ANN
    3. Categorical outcome. Binary as an option.
    4. Missing values in several features. Imputation not possible
    5. Naive Bayes is a decent choice for both accuracy and the amount of effort required. Similar to XGBoost for MIC, another big advantage with Naive Bayes is that it is extremely fast and requires a lot less computing resources when compared to ANN.
    6. Naive Bayes for Binary predictions is a lot more accurate than for categorical predictions.
    7. Again similar to the MIC case, ANN are the better choice for those who have the skills, who can spend the time and effort.

We can see that each dataset can be analysed using several different algorithms and the type of outcome desired (Binary or Categorical) with differing levels of accuracy in predictions.

We have also seen how we can handle missing values in our data. In the HFP dataset, we had no missing values. In the MIC dataset, we used imputation for filling in missing values and for the d130 dataset, we used a common label to represent missing values and went ahead with what was available. In both the MIC and d130 datasets, we judiciously chose what to do with missing values that could neither be imputed nor replaced with a label that made sense.

A common theme across all the 3 datasets has been ANN. ANN are extremely popular with ML learners, developers and enthusiasts because of their flexible and programmable nature. As we have seen, the enthusiasm of the ML community for ANN is not misplaced.

ANN are very flexible and can be built for the task at hand, but an important thought to bear in mind is that the ANN are suited to the dataset that they are built for and often cannot be reused as-built for a different dataset. This makes their extension and maintenance a costly proposition in terms of time and efforts. ANN also require plenty of computing resources.

For quick checks and also for those who do not yet have the skills, lack time or cannot expend the effort required to build ANN. Algorithms like Random Forest, Naive Bayes and XGBoost provide extremely good alternatives.

Based on what we have seen so far. we can use a simple common sense approach to answer the initial questions we had when we started off with the Project

1. How to assess the dataset and the associated complexity
    a. Starting to analyse data without understanding its basic characteristics and purpose will lead us astray.
    b. We can understand that complexity can stem from the number of features, the types of features, diversity in the feature characteristics, types of outcomes and number of outcomes among others.
    c. The first thing to do is look for a detailed dataset description to understand what the dataset contains and what the data is structured like. It not as glamorous as preparing and presenting reports but the menial work needs to be done.
    d. Verifying that the data matches the description is the next step. Sometimes dataset descriptions may not exactly match. An example to cite is that the d130 dataset has been in the wild since 2014 and the research article indicates that there are 24 features for the medications and so do most of the descriptions of the dataset. Actually there are only 23. Not that it matters much in this project but verification is necessary.
    e. Having the services of a Subject Matter Expert (SME) can greatly help us understand the data and how to process it. As an example, In the survival analysis of a Heart Failure Patient what exactly does Ejection Factor (EF) mean? How many different types of EF do we have? How important is it as a marker of the patient's survivability? If it is missing, can it be assumed or imputed somehow? If we just used statistical methods to impute the EF, does that really make sense? What about a Heart Attack Prediction? Which ECG parameters are the most important? These are questions that a SME can answer but the Data Analyst will struggle with.
2. How to get started with analysing the data
    a. When we need to get started with analysing the data, often the first question to ask is Who is looking for our analysis and what do they desire?
    b. Can we reduce the complexity of the outcome? As an example, For prediction of Lethal Events, is it required that we also accurately predict the category by which the event might occur? Is a binary prediction good enough? If categorisation is required, how accurately should we be able to predict the categories and the individual events within the categories
    c. we can often begin with a simple divide and conquer approach. We look at each feature to understand if it is needed? will it contribute to our analysis? can we look at keeping only what is important? Is some data missing? how does the missing data impact the analysis? Can we fill in the missing data somehow or do away with the feature?
    d. Using column names rather than column numbers is a good practice when dealing with complex datasets where the features are spread across. That way, we can create subsets and feed the algorithms with the features that make the most sense.
3. What kind of processing might be required
    a. Is the data ready to be processed as it is or do we need to perform some processing to get it into shape.
    b. Do we need processing steps like imputation, one-hot encoding and NA replacement among others.
    c. We also need to have a good understanding of how our Algorithms work and what tuning is required based on the data. Each Algorithm and its implementation has its own specifics and quirks. Random Forest cannot handle too many factors, Naive Bayes requires us to configure the features using the right variable types in addition to the Naive Bayes configuration, XGBoost and ANN can only work with numeric data. What does the ETA in XGBoost mean?
4. Which Algorithms to use to get some initial insight

    a. Simple answer, use the simplest ones like Naive Bayes or XGBoost first. Random Forest is a simple Algorithm too but can be computationally expensive for large datasets. Naive Bayes and XGBoost are not very computationally expensive and are extremely fast compared to other Algorithms.
    b. ANN are usually a bad choice for initial analysis as they need to be custom built for most tasks

5. Which ones to use for more detailed analysis.

    a. Difficult answer, we will get to know on a case by case basis.

First up, I must be honest in admitting that we have not made any great discovery but have arrived at some simple practices that can make our lives a little bit easier by providing a logical, methodical way to approach any Data Analysis project.

We can now conclude that the choice of Algorithms is to a large extent dependent on the data and the structure of the dataset. It is hard to know beforehand which Algorithm will perform best for any specific dataset & task combination. While many of the Algorithms can be tuned to suit the task at hand. It is wise to perform initial tests using simpler Algorithms before delving into more complex ones.

6 Conclusion

As we approach the end of this project and record the concluding notes, we can recollect our Project Approach.

Our main goals were to find public, unencumbered datasets around the topic of Cardiovascular Diseases (CVD) and build something practically useful around them.

We have seen how different datasets represent different aspects of CVD,

1. The Heart Failure Prediction (HFP) dataset represents the survival of patients affected by Heart Failure
2. The Myocardial Infarction Complications (MIC) dataset represents the survival of patients affected by Heart Attacks.
3. We modified the Diabetes 130-US Hospitals for Years 1999-2008 (d130) dataset to only extract details of patients whose primary diagnosis was CVD with Diabetes as secondary and additional secondary diagnoses.

When we look at any broad topic like CVD, we are faced with many challenges and just as many approaches to those challenges.

The first and foremost challenge is to find datasets that are current and ready for analysis. I have searched for and tried to obtain datasets from the WHO and several country government sites. Many of them are meant as health indicator reports and the results summarise the disease and fatality rates across different countries, provinces, states or even districts/counties based on Age, Ethnicity, Geographical Location, Lifestyle choices, Smoking, Alcohol Consumption among others.

Please see below for some examples.

https://www.cdc.gov/brfss/

https://chronicdata.cdc.gov/browse?category=Heart+Disease+%26+Stroke+Prevention&sortBy=relevance&pageSize=20

Though summarised data is readily available, finding raw data is quite difficult. In most places registration is required and conditional access is provided only for explicit research purposes with emphasis being placed on the same restrictions being applicable to other downstream recipients of the data.

There are several commercial vendors available who provide curated datasets that are ready for analysis at a price and with restrictions around usage and sharing of the data.

The next challenge is to do something with the dataset that has not been done before and can be considered worth the readers' and evaluators' time. This is just as difficult because many of the publicly available unencumbered datasets have been analysed and reports generated by several different organisations, researchers and students.

The other challenge that researchers face is that the R Language does not have a lot of native implementations of ANN packages. The current fully featured native implementation available is "torch for R" which is based on "pyTorch". Most of the larger ML community seems to use Python more than R.

My choice of Keras as API was more due to the smoother learning curve for Keras as compared to other ANN packages and because POSIT (the creators of RStudio and many open source R products) maintains the R Interface for Keras.

In terms of future work, something that I tried to do initially but could not successfully complete was to obtain a dataset that was current, complete and unencumbered, even with missing data, which could be cleaned up and be shared with the broader community. My first choice of dataset was one that could be used to diagnose early onset of diabetes and/or CVD or establish a causal link between them. The other choice was to look at the contributions to mortality by Infectious diseases like Tuberculosis among populations in Africa and South Asia. This is definitely something very useful if it can be done.

The other thing that can be done is use "torch for R" to build the ANN which is native to R and does not require Python in the background like Keras does.

I did not delve into feature engineering as it would elongate an already long project. I have just done enough to get the Algorithms to work and provide comparable results. Feature engineering can help increase accuracy and also reduce computing time and resources. Definitely something worth doing. The d130 dataset in particular has many categorical features that seem to increment the accuracy by very little and counteract each other.

For some reason, I could not get ADABoost and ADABag to provide sufficiently accurate results to include in the report. They also consumed more time and computing resources than the other Algorithms. They are both extremely capable, so worth exploring.

While this project does not help in the diagnosis of CVD any better, it provides students of Data Science researching CVD or other Medical Topics with some indicators towards the performance of various ML algorithms and the basic work required to make them work. They can build and improve upon what is available.

I will end the report with a note of thanks to the Readers and Evaluators for their time and patience and Everyone who has helped, directly and indirectly, in the Execution of the Project and the Creation of this Report.

References

1. Clore, J., Cios, K., DeShazo, J., & Strack, B. (2014). Diabetes 130-US Hospitals for Years 1999-2008 [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5230J.

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

3. Golovenkin, S., Shulman, V., Rossiev, D., Shesternya, P., Nikulina, S., Orlova, Y., & Voino-Yasenetsky, V. (2020). Myocardial infarction complications [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C53P5M.

4. Golovenkin, S.E.; Gorban, Alexander; Mirkes, Evgeny; Shulman, V.A.; Rossiev, D.A.; Shesternya, P.A.; et al. (2020). Myocardial infarction complications Database. University of Leicester. Dataset. https://doi.org/10.25392/leicester.data.12045261.v3

5. Joaquin Vanschoren and Jan N. van Rijn and Bernd Bischl and Luis Torgo. ***OpenML: networked science in machine learning.***SIGKDD Explorations 15(2), pp 49-60, 2013

6. Giuseppe Casalicchio and Jakob Bossek and Michel Lang and Dominik Kirchhoff and Pascal Kerschke and Benjamin Hofner and Heidi Seibold and Joaquin Vanschoren and Bernd Bischl. ***OpenML: An R package to connect to the machine learning platform OpenML.***Computational Statistics 32(3), pp 1-15, 2017

7. van Buuren, S., & Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, *45*(3), 1–67. https://doi.org/10.18637/jss.v045.i03

Appendix A - MIC Dataset Features & Outcomes

For accommodating differences in the Encoding and Analysis methods, we will produce a modified version of the description. We will order the descriptions of the Features and Targets as they appear in the dataset.

Table of abbreviations

FC is the functional class of angina pectoris in the last year

CHD is coronary heart disease.

HF is heart failure.

ECG is electrocardiogram.

AV is atrioventricular block.

LBBB is left bundle branch block.

RBBB is right bundle branch block.

QRS is QRS complex in ECG

IU is international unit.

ICU is intensive care unit.

ESR is erythrocyte sedimentation rate.

NSAID is non-steroidal anti-inflammatory drugs.

6.1 Attributes/Features

The dataset has 1700 observations and the following features. Not all observations have all features.

1. Record ID (ID): Unique identifier. Cannot be related to participant. It can be used for reference only.

Type : Numeric/ Integer

2. Age (AGE): Age of patient.

Type: Numeric/ Real.

Attribute	Min	Max	Mean	STD	Missing cases	Missing fraction
Age	26	92	61.86	11.26	8	0.47%

3. Gender (SEX):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	female	635	37.35%
1	male	1065	62.65%
	Missing		0 0%

4. Quantity of myocardial infarctions in the anamnesis (INF_ANAM):

Type: Ordinal/Categorical/Factor

Ordinal attribute can be interpreted as numerical. Possible encoding as is or cumulative dummy coding

Value	Represents	Cases	Fraction
0	zero	1060	62.35%
1	one	410	24.12%
2	two	147	8.65%
3	three and more	79	4.65%
	Missing	4	0.24%

5 .Exertional angina pectoris in the anamnesis (STENOK_AN):

Type:Ordinal/Categorical/Factor

Ordinal attribute can be interpreted as numerical. Possible encoding as is or cumulative dummy coding.

Value	Represents	Cases	Fraction
0	never	661	38.88%
1	during the last year	146	8.59%
2	one year ago	137	8.06%
3	two years ago	117	6.88%
4	three years ago	76	4.47%
5	4-5 years ago	125	7.35%
6	more than 5 years ago	332	19.53%
	Missing	106	6.24%

6. Functional class (FC) of angina pectoris in the last year (FK_STENOK):

Type: Ordinal/Categorical/Factor

Ordinal attribute. Possible usage of cumulative dummy coding.

Value	Represents	Cases	Fraction
0	there is no angina pectoris	661	38.88%
1	I FC	47	2.76%
2	II FC	854	50.24%
3	III FC	54	3.18%
4	IV FC	11	0.65%
	Missing	73	4.29%

7. Coronary heart disease (CHD) in recent weeks, days before admission to hospital (IBS_POST):

Type: Ordinal/Categorical/Factor

Ordinal attribute. Possible usage of cumulative dummy coding.

Value	Represents	Cases	Fraction
0	there was no CHD	418	24.59%
1	exertional angina pectoris	548	32.24%
2	unstable angina pectoris	683	40.18%
	Missing	51	3.00%

8. Heredity on CHD (IBS_NASL):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	Isn't burdened	45	2.65%
1	burdened	27	1.59%
	Missing	1628	95.76%

9. Presence of an essential hypertension (GB):

Type: Ordinal/Categorical/Factor

Ordinal attribute. Possible usage of cumulative dummy coding.

Value	Represents	Cases	Fraction
0	there is no essential hypertension	605	35.59%
1	Stage 1	11	0.65%
2	Stage 2	880	51.76%
3	Stage 3	195	11.47%
	Missing	9	0.53%

10. Symptomatic hypertension (SIM_GIPERT):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1635	96.18%
1	yes	57	3.35%
	Missing	8	0.47%

11. Duration of arterial hypertension (DLIT_AG):

Type: Ordinal/Categorical/Factor

Ordinal attribute. Possible usage of cumulative dummy coding.

Value	Represents	Cases	Fraction
0	there was no arterial hypertension	551	32.41%
1	one year	93	5.47%
2	two years	58	3.41%
3	three years	58	3.41%
4	four years	22	1.29%
5	five years	73	4.29%
6	6-10 years	165	9.71%
7	more than 10 years	432	25.41%
	Missing	248	14.59%

12. Presence of chronic Heart failure (HF) in the anamnesis (ZSN_A):

Type: Partially ordered attribute: Considered as Ordinal/Categorical/Factor

there are two lines of severities:

0<1<2<4, 0<1<3<4.

State 4 means simultaneous states 2 and 3

Possible usage of cumulative dummy coding.

Value	Represents	Cases	Fraction
0	there is no chronic heart failure	1468	86.35%
1	I stage	103	6.06%
2	II stage (heart failure due to right ventricular systolic dysfunction)	27	1.59%
3	III stage (heart failure due to left ventricular systolic dysfunction)	29	1.71%
4	IIB stage (heart failure due to left and right ventricular systolic dysfunction)	19	1.12%
	Missing	54	3.18%

13. Observing of arrhythmia in the anamnesis (nr_11):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1637	96.29%
1	yes	42	2.47%
	Missing	21	1.24%

14. Premature atrial contractions in the anamnesis (nr_01):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1675	98.53%
1	yes	4	0.24%
	Missing	21	1.24%

15. Premature ventricular contractions in the anamnesis (nr_02):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1660	97.65%
1	yes	19	1.12%
	Missing	21	1.24%

16. Paroxysms of atrial fibrillation in the anamnesis (nr_03):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1644	96.71%
1	yes	35	2.06%
	Missing	21	1.24%

17. A persistent form of atrial fibrillation in the anamnesis (nr_04):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1650	97.06%
1	yes	29	1.71%
	Missing	21	1.24%

18. Ventricular fibrillation in the anamnesis (nr_07):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1678	98.71%
1	yes	1	0.06%
	Missing	21	1.24%

19. Ventricular paroxysmal tachycardia in the anamnesis (nr_08):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1675	98.53%
1	yes	4	0.24%
	Missing	21	1.24%

20. First-degree AV block in the anamnesis (np_01):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1680	98.82%
1	yes	2	0.12%
	Missing	18	1.06%

21. Third-degree AV block in the anamnesis (np_04):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1679	98.76%
1	yes	3	0.18%
	Missing	18	1.06%

22. LBBB (anterior branch) in the anamnesis (np_05):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1671	98.29%
1	yes	11	0.65%
	Missing	18	1.06%

23. Incomplete LBBB in the anamnesis (np_07):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1681	98.88%
1	yes	1	0.06%
	Missing	18	1.06%

24. Complete LBBB in the anamnesis (np_08):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1676	98.59%
1	yes	6	0.35%
	Missing	18	1.06%

25. Incomplete RBBB in the anamnesis (np_09):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1680	98.82%
1	yes	2	0.12%
	Missing	18	1.06%

26. Complete RBBB in the anamnesis (np_10):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1679	98.76%
1	yes	3	0.18%
	Missing	18	1.06%

27. Diabetes mellitus in the anamnesis (endocr_01):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1461	85.94%
1	yes	228	13.41%
	Missing	11	0.65%

28. Obesity in the anamnesis (endocr_02):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1648	96.94%

1	yes	42	2.47%
	Missing	10	0.59%

29. Thyrotoxicosis in the anamnesis (endocr_03):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1677	98.65%
1	yes	13	0.76%
	Missing	10	0.59%

30. Chronic bronchitis in the anamnesis (zab_leg_01):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1559	91.71%
1	yes	134	7.88%
	Missing	7	0.41%

31. Obstructive chronic bronchitis in the anamnesis (zab_leg_02):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1572	92.47%
1	yes	121	7.12%
	Missing	7	0.41%

32. Bronchial asthma in the anamnesis (zab_leg_03):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1656	97.41%
1	yes	37	2.18%
	Missing	7	0.41%

33. Chronic pneumonia in the anamnesis (zab_leg_04):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1684	99.06%
1	yes	9	0.53%
	Missing	7	0.41%

34. Pulmonary tuberculosis in the anamnesis (zab_leg_06):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1684	99.06%
1	yes	9	0.53%
	Missing	7	0.41%

35. Systolic blood pressure according to Emergency Cardiology Team (S_AD_KBRIG) (mmHg):

Type: Numeric/Real

Attribute	Min	Max	Mean	STD	Missing cases	Missing fraction
S_AD_KBRIG	0	260	136.91	34.97	1076	63.29%

36. Diastolic blood pressure according to Emergency Cardiology Team (D_AD_KBRIG) (mmHg):

Type: Numeric/Real

Attribute	Min	Max	Mean	STD	Missing cases	Missing fraction
D_AD_KBRIG	0	190	81.39	19.73	1076	63.29%

37. Systolic blood pressure according to intensive care unit (S_AD_ORIT) (mmHg):

Type: Numeric/Real

Attribute	Min	Max	Mean	STD	Missing cases	Missing fraction
S_AD_ORIT	0	260	134.59	31.34	267	15.71%

38. Diastolic blood pressure according to intensive care unit (D_AD_ORIT) (mmHg):

Type: Numeric/Real

Attribute	Min	Max	Mean	STD	Missing cases	Missing fraction
D_AD_ORIT	0	190	82.75	18.31	267	15.71%

39. Pulmonary edema at the time of admission to intensive care unit (O_L_POST):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1578	92.82%
1	yes	110	6.47%
	Missing	12	0.71%

40. Cardiogenic shock at the time of admission to intensive care unit (K_SH_POST):

Type: Binary/Nominal

Value	Represents	Cases	Fraction

0	no	1639	96.41%
1	yes	46	2.71%
	Missing	15	0.88%

41. Paroxysms of atrial fibrillation at the time of admission to intensive care unit, (or at a pre-hospital stage) (MP_TP_POST):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1572	92.47%
1	yes	114	6.71%
	Missing	14	0.82%

42. Paroxysms of supraventricular tachycardia at the time of admission to intensive care unit, (or at a pre-hospital stage) (SVT_POST):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1680	98.82%
1	yes	8	0.47%
	Missing	12	0.71%

43. Paroxysms of ventricular tachycardia at the time of admission to intensive care unit, (or at a pre-hospital stage) (GT_POST):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1680	98.82%
1	yes	8	0.47%
	Missing	12	0.71%

44. Ventricular fibrillation at the time of admission to intensive care unit, (or at a pre-hospital stage) (FIB_G_POST):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1673	98.41%
1	yes	15	0.88%
	Missing	12	0.71%

45. Presence of an anterior myocardial infarction (left ventricular) (ECG changes in leads V1: V4) (ant_im):

Type: Ordinal/Categorical/Factor

Ordinal attribute can be interpreted as numerical. Possible encoding as is or cumulative dummy coding.

Value	Represents	Cases	Fraction
0	there is no infarct in this location	660	38.82%
1	QRS has no changes	392	23.06%
2	QRS is like QR-complex	39	2.29%
3	QRS is like Qr-complex	34	2.00%
4	QRS is like QS-complex	492	28.94%
	Missing	83	4.88%

46. Presence of a lateral myocardial infarction (left ventricular) (ECG changes in leads V5: V6 , I, AVL) (lat_im):

Type: Ordinal/Categorical/Factor

Ordinal attribute can be interpreted as numerical. Possible encoding as is or cumulative dummy coding.

Value	Represents	Cases	Fraction
0	there is no infarct in this location	576	33.88%
1	QRS has no changes	838	49.29%
2	QRS is like QR-complex	97	5.71%
3	QRS is like Qr-complex	72	4.24%
4	QRS is like QS-complex	37	2.18%
	Missing	80	4.71%

47. Presence of an inferior myocardial infarction (left ventricular) (ECG changes in leads III, AVF, II). (inf_im):

Type: Ordinal/Categorical/Factor

Ordinal attribute can be interpreted as numerical. Possible encoding as is or cumulative dummy coding.

Value	Represents	Cases	Fraction
0	there is no infarct in this location	937	55.12%
1	QRS has no changes	195	11.47%
2	QRS is like QR-complex	191	11.24%
3	QRS is like Qr-complex	121	7.12%
4	QRS is like QS-complex	176	10.35%
	Missing	80	4.71%

48. Presence of a posterior myocardial infarction (left ventricular) (ECG changes in V7: V9, reciprocity changes in leads V1:V3) (post_im):

Type: Ordinal/Categorical/Factor

Ordinal attribute can be interpreted as numerical. Possible encoding as is or cumulative dummy coding.

Value	Represents	Cases	Fraction
0	there is no infarct in this location	1370	80.59%
1	QRS has no changes	157	9.24%
2	QRS is like QR-complex	52	3.06%
3	QRS is like Qr-complex	35	2.06%
4	QRS is like QS-complex	14	0.82%
	Missing	72	4.24%

49. Presence of a right ventricular myocardial infarction (IM_PG_P):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1649	97.00%
1	yes	50	2.94%
	Missing	1	0.06%

50. ECG rhythm at the time of admission to hospital: sinus (with a heart rate 60-90) (ritm_ecg_p_01):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	519	30.53%
1	yes	1029	60.53%
	Missing	152	8.94%

51. ECG rhythm at the time of admission to hospital: atrial fibrillation (ritm_ecg_p_02):

Type Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1453	85.47%
1	yes	95	5.59%
	Missing	152	8.94%

52. ECG rhythm at the time of admission to hospital: atrial (ritm_ecg_p_04):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1525	89.71%
1	yes	23	1.35%
	Missing	152	8.94%

53. ECG rhythm at the time of admission to hospital: idioventricular (ritm_ecg_p_06):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1547	91.00%
1	yes	1	0.06%
	Missing	152	8.94%

54. ECG rhythm at the time of admission to hospital: sinus with a heart rate above 90 (tachycardia) (ritm_ecg_p_07):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1195	70.29%
1	yes	353	20.76%
	Missing	152	8.94%

55. ECG rhythm at the time of admission to hospital: sinus with a heart rate below 60 (bradycardia) (ritm_ecg_p_08):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1502	88.35%
1	yes	46	2.71%
	Missing	152	8.94%

56. Premature atrial contractions on ECG at the time of admission to hospital (n_r_ecg_p_01):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1527	89.82%
1	yes	58	3.41%
	Missing	115	6.76%

57. Frequent premature atrial contractions on ECG at the time of admission to hospital (n_r_ecg_p_02):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1577	92.76%
1	yes	8	0.47%
	Missing	115	6.76%

58. Premature ventricular contractions on ECG at the time of admission to hospital (n_r_ecg_p_03):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1381	81.24%
1	yes	204	12.00%
	Missing	115	6.76%

59. Frequent premature ventricular contractions on ECG at the time of admission to hospital (n_r_ecg_p_04):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1516	89.18%
1	yes	69	4.06%

	Missing	115	6.76%

60. Paroxysms of atrial fibrillation on ECG at the time of admission to hospital (n_r_ecg_p_05):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1515	89.12%
1	yes	70	4.12%
	Missing	115	6.76%

61. Persistent form of atrial fibrillation on ECG at the time of admission to hospital (n_r_ecg_p_06):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1553	91.35%
1	yes	32	1.88%
	Missing	115	6.76%

62. Paroxysms of supraventricular tachycardia on ECG at the time of admission to hospital (n_r_ecg_p_08):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1581	93.00%
1	yes	4	0.24%
	Missing	115	6.76%

63. Paroxysms of ventricular tachycardia on ECG at the time of admission to hospital (n_r_ecg_p_09):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1583	93.12%
1	yes	2	0.12%
	Missing	115	6.76%

64. Ventricular fibrillation on ECG at the time of admission to hospital (n_r_ecg_p_10):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1583	93.12%
1	yes	2	0.12%
	Missing	115	6.76%

65. Sinoatrial block on ECG at the time of admission to hospital (n_p_ecg_p_01):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1583	93.12%
1	yes	2	0.12%
	Missing	115	6.76%

66. First-degree AV block on ECG at the time of admission to hospital (n_p_ecg_p_03):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1553	91.35%
1	yes	32	1.88%
	Missing	115	6.76%

67. Type 1 Second-degree AV block (Mobitz I/ Wenckebach) on ECG at the time of admission to hospital (n_p_ecg_p_04):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1580	92.94%
1	yes	5	0.29%
	Missing	115	6.76%

68. Type 2 Second-degree AV block (Mobitz II/Hay) on ECG at the time of admission to hospital (n_p_ecg_p_05):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1583	93.12%
1	yes	2	0.12%
	Missing	115	6.76%

69. Third-degree AV block on ECG at the time of admission to hospital (n_p_ecg_p_06):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1558	91.65%
1	yes	27	1.59%
	Missing	115	6.76%

70. LBBB (anterior branch) on ECG at the time of admission to hospital (n_p_ecg_p_07):

Type: Binary/Nominal

Value	Represents	Cases	Fraction

0	no	1483	87.24%
1	yes	102	6.00%
	Missing	115	6.76%

71. LBBB (posterior branch) on ECG at the time of admission to hospital (n_p_ecg_p_08):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1578	92.82%
1	yes	7	0.41%
	Missing	115	6.76%

72. Incomplete LBBB on ECG at the time of admission to hospital (n_p_ecg_p_09):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1575	92.65%
1	yes	10	0.59%
	Missing	115	6.76%

73. Complete LBBB on ECG at the time of admission to hospital (n_p_ecg_p_10):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1551	91.24%
1	yes	34	2.00%
	Missing	115	6.76%

74. Incomplete RBBB on ECG at the time of admission to hospital (n_p_ecg_p_11):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1557	91.59%
1	yes	28	1.65%
	Missing	115	6.76%

75. Complete RBBB on ECG at the time of admission to hospital (n_p_ecg_p_12):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1507	88.65%
1	yes	78	4.59%
	Missing	115	6.76%

76. Fibrinolytic therapy by Celiasum 750k IU (fibr_ter_01): Celiasum

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1677	98.65%
1	yes	13	0.76%
	Missing	10	0.59%

77. Fibrinolytic therapy by Celiasum 1m IU (fibr_ter_02):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1674	98.47%
1	yes	16	0.94%
	Missing	10	0.59%

78. Fibrinolytic therapy by Celiasum 3m IU (fibr_ter_03):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1622	95.41%
1	yes	68	4.00%
	Missing	10	0.59%

79. Fibrinolytic therapy by Streptase (fibr_ter_05):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1686	99.18%
1	yes	4	0.24%
	Missing	10	0.59%

80. Fibrinolytic therapy by Celiasum 500k IU (fibr_ter_06):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1681	98.88%
1	yes	9	0.53%
	Missing	10	0.59%

81. Fibrinolytic therapy by Celiasum 250k IU (fibr_ter_07):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1684	99.06%
1	yes	6	0.35%

	Missing	10	0.59%

82. Fibrinolytic therapy by Streptodecase 1.5m IU (fibr_ter_08):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1688	99.29%
1	yes	2	0.12%
	Missing	10	0.59%

83. Hypokalemia ($<$ 4 mmol/L) (GIPO_K):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	797	46.88%
1	yes	534	31.41%
	Missing	369	21.71%

84. Serum potassium content (K_BLOOD) (mmol/L):

Type: Numeric/Real

Attribute	Min	Max	Mean	STD	Missing cases	Missing fraction
K_BLOOD	2.3	8.2	4.19	0.75	371	21.82%

85. Increase of sodium in serum (more than 150 mmol/L) (GIPER_Na):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1295	76.18%
1	yes	30	1.76%
	Missing	375	22.06%

86. Serum sodium content (NA_BLOOD) (mmol/L):

Type: Numeric/Real

Attribute	Min	Max	Mean	STD	Missing cases	Missing fraction
NA_BLOOD	117	169	136.55	6.51	375	22.06%

87. Serum AlAT content (ALT_BLOOD) (IU/L):

Type: Numeric/Real

Attribute	Min	Max	Mean	STD	Missing cases	Missing fraction
ALT_BLOOD	0.03	3	0.48	0.39	284	16.71%

88. Serum AsAT content (AST_BLOOD) (IU/L):

Type: Numeric/Real

Attribute	Min	Max	Mean	STD	Missing cases	Missing fraction
AST_BLOOD	0.04	2.15	0.26	0.2	285	16.67%

89. Serum CPK content (KFK_BLOOD) (IU/L):

Type: Numeric/Real

Attribute	Min	Max	Mean	STD	Missing cases	Missing fraction
KFK_BLOOD	1.2	3.6	2	0.95	1696	99.76%

*Note: There are too many missing values for us to make any sense of how the feature affects the outcomes. We will not consider this feature in our Analysis.

90. White blood cell count (billions per liter) (L_BLOOD):

Type: Numeric/Real

Attribute	Min	Max	Mean	STD	Missing cases	Missing fraction
L_BLOOD	2	27.9	8.78	3.40	125	7.35%

91. ESR (Erythrocyte sedimentation rate) (ROE) (mm):

Type: Numeric/Real

Attribute	Min	Max	Mean	STD	Missing cases	Missing fraction
ROE	1	140	13.44	11.29	203	19.94%

92. Time elapsed from the beginning of the attack of CHD to the hospital (TIME_B_S):

Type: Ordinal/Categorical/Factor

Ordinal attribute. Possible usage of cumulative dummy coding.

Value	Represents	Cases	Fraction
1	less than 2 hours	198	11.65%
2	2-4 hours	360	21.18%
3	4-6 hours	175	10.29%
4	6-8 hours	87	5.12%
5	8-12 hours	92	5.41%
6	12-24 hours	151	8.88%
7	more than 1 days	141	8.29%
8	more than 2 days	101	5.94%
9	more than 3 days	269	15.82%
	Missing	126	7.41%

93. Relapse of the pain in the first hours of the hospital period (R_AB_1_n):

Type: Ordinal/Categorical/Factor

Ordinal attribute can be interpreted as numerical. Possible encoding as is or cumulative dummy coding.

Value	Represents	Cases	Fraction
0	there is no relapse	1282	75.41%
1	only one	298	17.53%
2	2 times	78	4.59%
3	3 or more times	26	1.53%
	Missing	16	0.94%

94. Relapse of the pain in the second day of the hospital period (R_AB_2_n):

Type: Ordinal/Categorical/Factor

Ordinal attribute can be interpreted as numerical. Possible encoding as is or cumulative dummy coding.

Value	Represents	Cases	Fraction
0	there is no relapse	1414	83.18%
1	only one	133	7.82%
2	2 times	44	2.59%
3	3 or more times	1	0.06%
	Missing	108	6.35%

95. Relapse of the pain in the third day of the hospital period (R_AB_3_n):

Type: Ordinal/Categorical/Factor

Ordinal attribute can be interpreted as numerical. Possible encoding as is or cumulative dummy coding.

Value	Represents	Cases	Fraction
0	there is no relapse	1469	86.41%
1	only one	86	5.06%
2	2 times	15	0.88%
3	3 or more times	2	0.12%
	Missing	128	7.53% (Incorrectly Marked in Original)

*Note: There is a very small Typographic Error in the Original Document in the “Missing” value. It has been corrected here.

96. Use of opioid drugs by the Emergency Cardiology Team (NA_KB):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	425	25.00%
1	yes	618	36.35%
	Missing	657	38.65%

97. Use of NSAIDs by the Emergency Cardiology Team (NOT_NA_KB):

Type: Binary/Nominal

Value	Represents	Cases	Fraction

0	no	313	18.41%
1	yes	701	41.23%
	Missing	375	22.06%

98.Use of lidocaine by the Emergency Cardiology Team (LID_KB):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	627	36.88%
1	yes	396	23.29%
	Missing	677	39.82%

99. Use of liquid nitrates in the ICU (NITR_S):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1496	88.00%
1	yes	195	11.47%
	Missing	9	0.53%

100. Use of opioid drugs in the ICU in the first hours of the hospital period (NA_R_1_n):

Type: Ordinal/Categorical/Factor

Ordinal attribute can be interpreted as numerical. Possible encoding as is or cumulative dummy coding.

Value	Represents	Cases	Fraction
0	no	1108	65.18%
1	once	409	24.06%
2	twice	132	7.76%
3	three times	35	2.06%
4	four times	11	0.65%
	Missing	5	0.29%

101. Use of opioid drugs in the ICU in the second day of the hospital period (NA_R_2_n):

Type: Ordinal/Categorical/Factor

Ordinal attribute can be interpreted as numerical. Possible encoding as is or cumulative dummy coding.

Value	Represents	Cases	Fraction
0	no	1474	86.71%
1	once	87	5.12%
2	twice	30	1.76%
3	three times	1	0.06%
	Missing	108	6.35%

102. Use of opioid drugs in the ICU in the third day of the hospital period (NA_R_3_n):

Type: Ordinal/Categorical/Factor

Ordinal attribute can be interpreted as numerical. Possible encoding as is or cumulative dummy coding.

Value	Represents	Cases	Fraction
0	no	1493	87.82%
1	once	60	3.53%
2	twice	16	0.94%
	Missing	131	7.71%

103. Use of NSAIDs in the ICU in the first hours of the hospital period (NOT_NA_1_n):

Type: Ordinal/Categorical/Factor

Ordinal attribute can be interpreted as numerical. Possible encoding as is or cumulative dummy coding.

Value	Represents	Cases	Fraction
0	no	1237	72.76%
1	once	376	22.12%
2	twice	53	3.12%
3	three times	17	1.00%
4	four or more times	7	0.41%
	Missing	10	0.59%

104. Use of NSAIDs in the ICU in the second day of the hospital period (NOT_NA_2_n):

Type: Ordinal/Categorical/Factor

Ordinal attribute can be interpreted as numerical. Possible encoding as is or cumulative dummy coding.

Value	Represents	Cases	Fraction
0	no	1454	85.53%
1	once	95	5.59%
2	twice	38	2.24%
3	three times	3	0.18%
	Missing	110	6.47%

105. Use of NSAIDs in the ICU in the third day of the hospital period (NOT_NA_3_n):

Type: Ordinal/Categorical/Factor

Ordinal attribute can be interpreted as numerical. Possible encoding as is or cumulative dummy coding.

Value	Represents	Cases	Fraction
0	no	1474	86.71%
1	once	57	3.35%
2	twice	38	2.24%
	Missing	131	7.71%

106. Use of lidocaine in the ICU (LID_S_n):

Type: Binary/Nominal

Value	Represents	Cases	Fraction

0	no	1211	71.24%
1	yes	479	28.18%
	Missing	10	0.59%

107. Use of beta-blockers in the ICU (B_BLOK_S_n):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1474	86.71%
1	yes	215	12.65%
	Missing	11	0.65%

108. Use of calcium channel blockers in the ICU (ANT_CA_S_n):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	562	33.06%
1	yes	1125	66.18%
	Missing	13	0.76%

109. Use of anticoagulants (heparin) in the ICU (GEPAR_S_n):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	480	28.24%
1	yes	1203	70.76%
	Missing	17	1.00%

110. Use of acetylsalicylic acid in the ICU (ASP_S_n):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	431	25.35%
1	yes	1252	73.65%
	Missing	17	1.00%

111. Use of Ticlid in the ICU (TIKL_S_n):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1654	97.29%
1	yes	30	1.76%
	Missing	16	0.94%

112. Use of Trental in the ICU (TRENT_S_n):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1343	79.00%
1	yes	341	20.06%
	Missing	16	0.94%

6.2 Complications

113. Atrial fibrillation (FIBR_PREDS):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1530	90.00%
1	yes	170	10.00%
	Missing		0.0%

114. Supraventricular tachycardia (PREDS_TAH):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1680	98.82%
1	yes	20	1.18%
	Missing		0.0%

115. Ventricular tachycardia (JELUD_TAH):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1658	97.53%
1	yes	42	2.47%
	Missing		0.0%

116. Ventricular fibrillation (FIBR_JELUD):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1629	95.82%
1	yes	71	4.18%
	Missing		0.0%

117. Third-degree AV block (A_V_BLOK):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1643	96.65%
1	yes	57	3.35%
	Missing		0.0%

118. Pulmonary edema (OTEK_LANC):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1541	90.65%
1	yes	159	9.35%
	Missing		0.0%

119. Myocardial rupture (RAZRIV):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1646	96.82%
1	yes	54	3.18%
	Missing		0.0%

120. Dressler syndrome (DRESSLER):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1625	95.59%
1	yes	75	4.41%
	Missing		0.0%

121. Chronic heart failure (ZSN):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1306	76.82%
1	yes	394	23.18%
	Missing		0.0%

122. Relapse of the myocardial infarction (REC_IM):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1541	90.65%
1	yes	159	9.35%
	Missing		0.0%

123. Post-infarction angina (P_IM_STEN):

Type: Binary/Nominal

Value	Represents	Cases	Fraction
0	no	1552	91.29%
1	yes	148	8.71%
	Missing		0.0%

124. Lethal outcome (cause) (LET_IS):

Type: Category (Not Ordered): Considered as Categorical/Factor

Value	Represents	Cases	Fraction
0	unknown (alive)	1429	84.06%
1	cardiogenic shock	110	6.47%
2	pulmonary edema	18	1.06%
3	myocardial rupture	54	3.18%
4	progress of congestive heart failure	23	1.35%
5	thromboembolism	12	0.71%
6	asystole	27	1.59%
7	ventricular fibrillation	27	1.59%
	Missing		0.0%