Advanced Data Analysis - Group 6

Jinhao Wang, Jeffrey Wong, Nicolaus Wong, Rui Zhu

April 2021

1 Introduction

This chapter contains an overview of Hepatitis C Virus, dataset introduction and objectives for the project.

1.1 Hepatitis C Virus

Hepatitis C Virus is an infection that causes serious damage to the liver. The virus is spread through the infected individual's blood or body fluids.[1] There are two types and four stages of HCV. The two types of HCV are fibrosis, which is the scarring of the tissue in the liver and cirrhosis, which is a late-stage scarring of the liver. The four stages of HCV are:

Stage 1: mild fibrosis without walls of scarring.
Stage 2: mild to moderate fibrosis with walls of scarring.
Stage 3: fibrosis with scarring that has spread to different parts of the liver.
Stage 4: severe scarring, leading to cirrhosis.

Although there are medical procedures to detect the presence of HCV, the purpose of this report is to use machine learning techniques to create a preliminary diagnosis for the likelihood of having HCV.

1.2 Dataset Introduction

The HCV dataset was created by Ralf Lichtinghagen, Frank Klawonn, Georg Hoffmann with the goal of categorizing blood donors vs HCV individuals. It contains a list of 615 blood donors, where 82 patients have HCV within their bodily fluids. Each observation consists of 12 variables: age, sex, albumin levels (ALB), alkaline phosphatase levels (ALP), Alanine Aminotransferase levels (ALT), Aspartate Aminotransferase levels (AST), basic insulation levels (BIL), cholinesterase levels (CHE), cholesterol levels (CHOL), creatinine levels (CREA), Gamma-Glutamyl Transferase levels (GGT) and protein levels (PROT).

The objective of the analysis of the dataset is to determine which variables are significant in determining the presence of HCV. The dataset consists of a variety of patients, thus there may be some outliers, meaning that outlier detection processes could be implemented to develop a more accurate model. Furthermore, the dataset also consists of a categorical variable of sex, thus, creating dummy variables for sex will be required to analyze the impact of sex on HCV.

1.3 Objectives

The dataset has given us complete information on the 12 parameters used to determine HCV. However, there may be times where incomplete data will be submitted when it comes to testing for HCV. The project will be to determine the most important parameters for detecting the presence of HCV and comparing the performance between different classification methods.

2 Methodology

In this section, we will be describing the methods used in both exploratory and main data analyses. This includes Generalized Linear Mixed Model (GLMM), Random Forest, and neural network.

2.1 Generalized Linear Mixed Model

As we have learnt in the GLM course, Generalized Linear Model is a general framework that was proposed by Nelder and Wedderburn (1972) using common features shared by different linear models. This includes linear regression and analysis-of-variance models, logit and probit models for quantal responses, log-linear models and multinomial response models for counts and some commonly used models for survival data.[2] A generalized linear model follows two components:

1. The response should follow a distribution from the exponential family of distributions.
2. The link function describes how the mean of the response and a linear combination of the predictors are related.

Generalized Linear Mixed Model is an extension of a Generalized linear model that involves models with random terms in the linear predictor.[3] It can also be seen as an extension of linear mixed models to allow response variables from different distributions, such as binary responses. In our case, we are classifying patients from Blood Donor, suspect Blood Donor, Hepatitis, Fibrosis and Cirrhosis and categorizing them to HCV and not HCV, in which the responses might not be normally distributed. Therefore, GLMM could be an excellent fit for us.

The general form of the model is:

$$\mathbf{y} = X\beta + Zu + \varepsilon$$

Where $\mathbf{y}$ is a $N \times 1$ column vector, the outcome variable; $\mathbf{X}$ is a $N \times p$ matrix of the p predictor variables; β is a $p \times 1$ column vector of the fixed-effects regression coefficients; $\mathbf{Z}$ is the $N \times q$ design matrix for the q random effects (the random complement to the fixed $\mathbf{X}$); u is a $q \times 1$ vector of the random effects (the random complement to the fixed β); and ε is a $N \times 1$ column vector of the residuals, that part of $\mathbf{y}$ that is not explained by the model, $X\beta + Zu$.[4]

2.2 Random Forest

Random forest is a supervised classification and regression algorithm, providing a better way to eliminate the significant predicament of accuracy optimization and over-fitting[5]. It is also a non-parametric algorithm, meaning there are no formal assumptions, thus can handle skewed

and multi-modal data as well as categorical data that are ordinal or non-ordinal. Since the HCV dataset contains binary categorical data, the random forest algorithm is a valid method in this case.

The random forest algorithm improves the bagged trees by letting each partition consider only a subset of the predictor variables. Therefore, when we have a large number of correlated predictor variables or a very strong predictor variable, it is usually useful to construct a random forest with a small value of $m \approx \sqrt{p}$ to build a random forest[6], where p is the total number of predictors and m is the number of selected parameters in each subset. In this way, the correlation between predictor variables will be eliminated as much as possible, and the strong predictor variable will not be considered in $(p-m)/p$ splits, thus decorrelates the trees and makes the result more reliable. What's more, due to the strong law of large numbers, random forests will always converge. That being said, as more trees are used, random forests will not suffer from over-adaptation.

In random forest, Gini impurity is a computationally efficient approximation to the entropy. It measures how well a potential split is separating the samples of the two classes in this specific feature.

Gini impurity is calculated as

$$i(\tau) = 1 - p_1^2 - p_0^2$$

Gini importance indicates how often a particular feature θ was selected for a split, and how large its overall discriminative value was for the classification problem under study. It can be calculated as

$$I_G = \sum_T \sum_\tau \Delta i_\theta(\tau, T)$$

Where Δi is the decrease of $i(\tau)$ and is defined as $\Delta i(\tau) = i(\tau) - p_l i\left(\tau_l\right) - p_r i\left(\tau_r\right)$.[7]

In R, the random forest algorithm can be accessed by the package randomForest, the number of trees to grow and the number of features used in the construction of each tree need to be specified.

2.3 Neural Network

Neural network (or the "vanilla" neural set) is a two-stage regression or classification model. The main idea of neural network is to extract linear combinations of the inputs as derived features and then model the response as a nonlinear function of these features[8]. Derived features are denoted by the term Z_m, and it is acquired from the linear combination of predictors. We then model the response Y_k as a function of linear combinations of the derived features, where m is the number of derived features after processing the predictors X and k is the number of classes in the response.

$$Z_m = \sigma\left(\alpha_{0m} + \alpha_m^T X\right), m = 1, \ldots, M$$

$$T_k = \beta_{0k} + \beta_k^T Z, k = 1, \ldots, K$$

$$f_k(X) = g_k(T), k = 1, \ldots, K$$

Here, $\sigma(v)$ represents the activation function, and we have chosen the Rectified Linear Unit (ReLU) as our activation function, ReLU function is defined as

$$f(x) = \max(0, x)$$

Where x is the input to a neuron.

The advantages of ReLU that make it superior to sigmoid activation function include model sparsity: ReLU's ability to turn negative inputs to zero values can significantly accelerates the model and avoids overfitting, and causes the network to be sparse. Model sparsity ensures that neurons are processing meaningful aspects of the problem; ReLU is also linear comparing to non-linear functions such as Sigmoid and TanH, which makes it computationally cheaper and it converges faster, it doesn't have the vanishing gradient problem suffered by other non-linear activation functions.

Given we are dealing with a binary classification problem, our output layer contains one node that will predict the probability of a patient having HCV or not.

3 Exploratory Data Analysis

Before we dive into the main data analysis, we need to perform an exploratory analysis on the data so that we can understand the data that we are dealing with. Looking at Figure 1, we can see that there are 615 observations with 14 features. However, feature X seems to be a numbering system for each row. We can also see that every feature except for Category and Sex is numerical. Let's take a closer look at some of these features to better understand the data.

```
[1] "Quick peek at the Data"
'data.frame':   615 obs. of  14 variables:
 $ X       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Category: chr  "0=Blood Donor" "0=Blood Donor" "0=Blood Donor" "0=Blood Donor" ...
 $ Age     : int  32 32 32 32 32 32 32 32 32 32 ...
 $ Sex     : chr  "m" "m" "m" "m" ...
 $ ALB     : num  38.5 38.5 46.9 43.2 39.2 41.6 46.3 42.2 50.9 42.4 ...
 $ ALP     : num  52.5 70.3 74.7 52 74.1 43.3 41.3 41.9 65.5 86.3 ...
 $ ALT     : num  7.7 18 36.2 30.6 32.6 18.5 17.5 35.8 23.2 20.3 ...
 $ AST     : num  22.1 24.7 52.6 22.6 24.8 19.7 17.8 31.1 21.2 20 ...
 $ BIL     : num  7.5 3.9 6.1 18.9 9.6 12.3 8.5 16.1 6.9 35.2 ...
 $ CHE     : num  6.93 11.17 8.84 7.33 9.15 ...
 $ CHOL    : num  3.23 4.8 5.2 4.74 4.32 6.05 4.79 4.6 4.1 4.45 ...
 $ CREA    : num  106 74 86 80 76 111 70 109 83 81 ...
 $ GGT     : num  12.1 15.6 33.2 33.8 29.9 91 16.9 21.5 13.7 15.9 ...
 $ PROT    : num  69 76.5 79.3 75.7 68.7 74 74.5 67.1 71.3 69.9 ...
```

Figure 1: Quick Peek at the Data

```
       X            Category              Age              Sex                 ALB              ALP              ALT              AST              BIL        
 Min.   :  1.0   Length:615         Min.   :19.00   Length:615         Min.   :14.90   Min.   : 11.30   Min.   :  0.90   Min.   : 10.60   Min.   :  0.8  
 1st Qu.:154.5   Class :character   1st Qu.:39.00   Class :character   1st Qu.:38.80   1st Qu.: 52.50   1st Qu.: 16.40   1st Qu.: 21.60   1st Qu.:  5.3  
 Median :308.0   Mode  :character   Median :47.00   Mode  :character   Median :41.95   Median : 66.20   Median : 23.00   Median : 25.90   Median :  7.3  
 Mean   :308.0                      Mean   :47.41                      Mean   :41.62   Mean   : 68.28   Mean   : 28.45   Mean   : 34.79   Mean   : 11.4  
 3rd Qu.:461.5                      3rd Qu.:54.00                      3rd Qu.:45.20   3rd Qu.: 80.10   3rd Qu.: 33.08   3rd Qu.: 32.90   3rd Qu.: 11.2  
 Max.   :615.0                      Max.   :77.00                      Max.   :82.20   Max.   :416.60   Max.   :325.30   Max.   :324.00   Max.   :254.0  
                                                                       NA's   :1       NA's   :18       NA's   :1                                        
      CHE              CHOL             CREA             GGT              PROT       
 Min.   : 1.420   Min.   : 1.430   Min.   :   8.00   Min.   :  4.50   Min.   :44.80  
 1st Qu.: 6.935   1st Qu.: 4.610   1st Qu.:  67.00   1st Qu.: 15.70   1st Qu.:69.30  
 Median : 8.260   Median : 5.300   Median :  77.00   Median : 23.30   Median :72.20  
 Mean   : 8.197   Mean   : 5.368   Mean   :  81.29   Mean   : 39.53   Mean   :72.04  
 3rd Qu.: 9.590   3rd Qu.: 6.060   3rd Qu.:  88.00   3rd Qu.: 40.20   3rd Qu.:75.40  
 Max.   :16.410   Max.   : 9.670   Max.   :1079.10   Max.   :650.90   Max.   :90.00  
                  NA's   :10                                          NA's   :1      
```

Figure 2: Mean, NA and Interquartile Range

Looking at the mean and interquartile ranges of each feature, in Figure 2, we can see that the average person in this data set is aged 47 years old, with the minimum and maximum being 19 and 77 years old respectively. Looking at Figure 12, we can see that the distribution looks almost normal, except for some light Positive Skew, which is also indicated by the median age being slightly smaller than the mean age. Taking a look at the ages across the gender, in Figure 13, we can see that the average male is slightly younger than the average female, with both being aged in the upper 40s, and the males having a larger interquartile range compared to the females. We can see from looking at Figure 5 that there were more male observations, with 377 males and 238 females, with the males having more cases of having the 3 more severe classifications of Hepatitis, even in proportion to their bigger observation size.

Figure 3: Distribution of Age

Figure 4: Box-Plot of Gender Interquartile Range Across Ages

Sex	0=Blood Donor	0s=suspect Blood Donor	1=Hepatitis	2=Fibrosis	3=Cirrhosis
f	215	1	4	8	10
m	318	6	20	13	20

Figure 5: Category Count over Gender

Next, if we look at the Category feature in the data, we can see from Figure 6 that most of our observations were in the category of "0=Blood Donors", followed by "3=Cirhosis", the most severe case of Hepatitis C. Taking a further dive into the division of Categories by age in Figure 7, we can see that the average classification of Cirrhosis happens in patients aged in their fifties, with the earliest cases happening in their late thirties, while the less severe case of Fibrosis is mostly

concentrated around the early fifties, with earliest cases happening prior to a patient's 30s. Finally, we can see that the classification of Hepatitis, it ranges from the twenties, all the way to the early sixties, with most falling in between early thirties to late forties. Seeing as most cases start sometime in the thirties, it is advised that patients and those with access to doctors do checkups around 30 years old, and again around 35 and 40 to catch these cases early, seeing as more mature cases of Hepatitis, like Cirrhosis and Fibrosis, tend to mature and materialize in the later thirties to early forties.

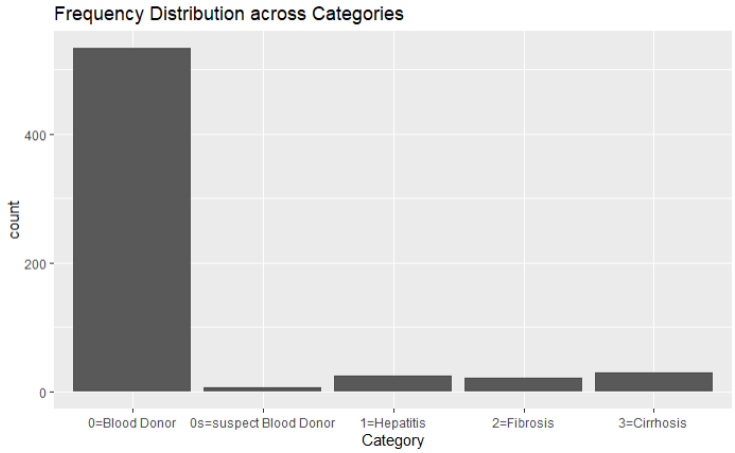


Figure 6: Distribution of the Categories Classification

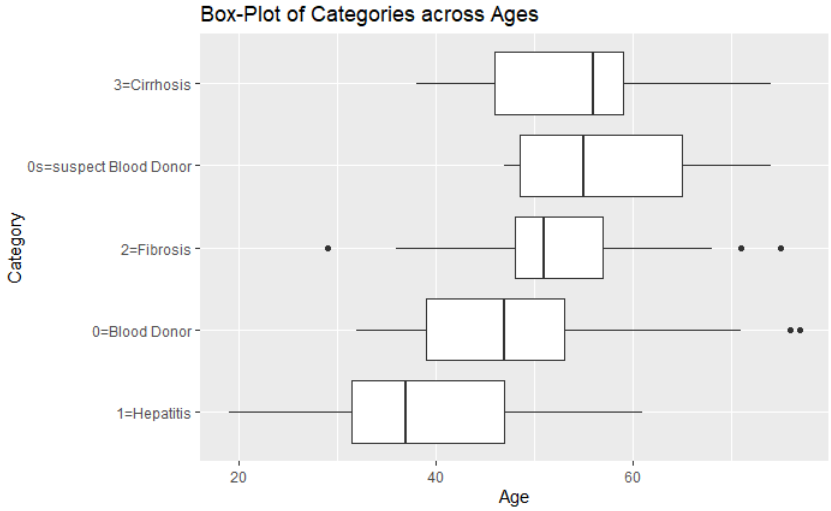


Figure 7: Box-Plot of Category Classifications Interquartile Range across Ages

Looking at Figure 2 and the histograms in the appendix, we can also get an understanding of the distribution, mean and interquartile ranges of the rest of the features, as well as the skewness found in the observations of the data; Albumin levels (ALB), Alkaline Phosphatase levels (ALP), Alanine Aminotransferase levels (ALT), Aspartate Aminotransferase levels (AST), Basic Insulation levels (BIL), Cholinesterase levels (CHE), Cholesterol levels (CHOL), Creatinine levels (CREA), Gamma-Glutamyl Transferase levels (GGT) and Protein levels (PROT). We've left the plots and analysis in Section 1 of the Appendix due to space constraints in this paper.

4 Main Data Analysis

4.1 Evaluations for Classification

This section is intended to introduce how methods are applied in the main analysis. Since we are currently only interested in the appearance of HCV patients, outcomes are separated into a binary category before implementation, while 1 means positive for HCV and 0 for donors regardless of regular or not (suspect). Figure 8 displays the count for each outcome.

The dataset is randomly split into training and testing sets at a percentage of 7:3.

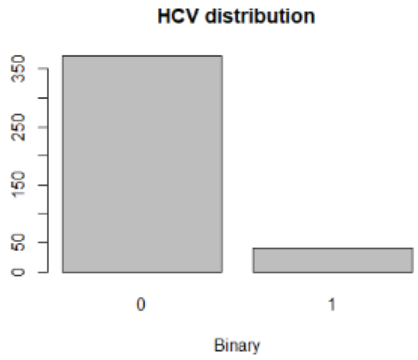


Figure 8: Whether or not positive for HCV

4.2 Random Forest

Due to the lack of the measurements made on independent variables for each continuous observation, we choose not to apply the linear discriminant analysis. Instead, we use the random forest to empirically ensemble a model composed with decision trees of predictors using R default bootstrapping.

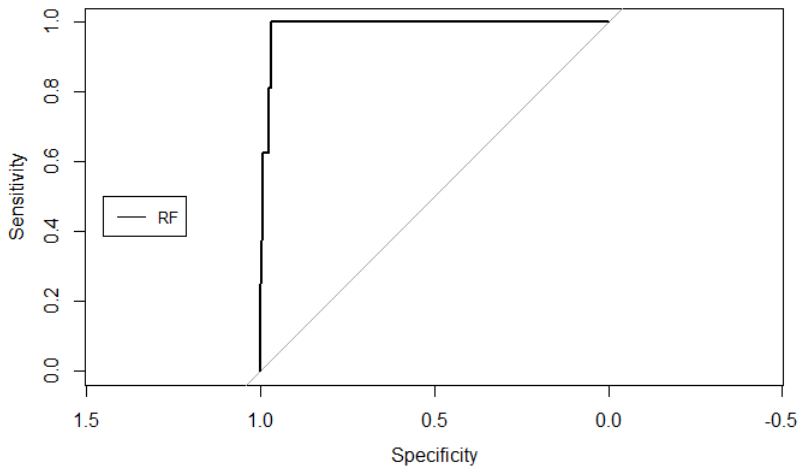


Figure 9: ROC plot for Random forest of test set

The random forest algorithm is applied to our binary outcome by fitting a model without the outcome Category. We set the number of trees to 10 with an additional tree node for the uncategorized outcomes and maximum of a 100 nodes to regulate our RF model without normalizing the vote counts.

	threshold	specificity	sensitivity	accuracy	precision
Random Forest	0.2833333	0.9689441	1	0.9717514	0.7619048

Table 1: Evaluations for binary classification using Random Forest

Through the output of the coordinates of the ROC curve at the best threshold, we figure random forest provides a relatively low threshold when classifying the observation. However, its performance is excellent with perfect sensitivity.

4.2.1 Gini Importance

Notably, in R, the random forest ROC curve in Figure 9 shows that it has the best performance compared to other methods that could potentially be used. we can infer that the predictors are significant to explain out model. The node impurity computed by Gini index in Table 2 also shows the importance of each predictor.

Interpretationally, the top 3 most important features are Aspartate Aminotransferase levels (AST), Alanine Aminotransferase levels (ALT), Alkaline phosphatase levels (ALP) respectively. The scientific way is choosing the most, the least and medium important predictors testing its effect of Gini importance. In the method of predictor selection. The logarithm formula is used for value-centering. In this case, AST, CHE and Sex are chosen for future potential use. However, we should be aware that Gini importance can be viewed as an overfitting problem: we compute the loss reduction on the out-of-bag instead of the in-bag training samples. [9]

	NodePurity	logNodePurity
Age	3.610843e-01	-1.0186438
Sex	2.775558e-18	-40.4256800
ALB	4.248468e-01	-0.8560265
ALP	3.345247e+00	1.2075404
ALT	7.598513e+00	2.0279526
AST	1.973172e+01	2.9822275
BIL	1.716077e+00	0.5400411
CHE	1.254098e+00	0.2264169
CHOL	3.677537e-01	-1.0003418
CREA	4.640634e-01	-0.7677341
GGT	2.672794e+00	0.9831243
PROT	6.762646e-15	-32.6273622

Table 2: Variable Importance Measure of Random Forest Model

4.3 Generalized Linear Mixed Model

The GLMM model fitted with multivariate normal random effects, using Penalized Quasi-maximum likelihood. When setting a feature as the random effect, we are considering more about the distribution of this feature, instead of its estimated values. To verify the credibility of Gini importance provided by the random forest algorithm and to test the pros and cons between models, we will be applying these features as the random effects of GLMM models.

We choose the optimal predictors from our random forest model, as well as the median to build our GLMM models. Using AST (maximum), CHE (median) and Sex (minimum) as the random effect for our GLMM model respectively. Reliably, one of the models provides nearly perfect performance on the test set, that is, the importance of random forest is somehow trustworthy. 0.994 specificity, 0.938 sensitivity of Sex and an accuracy of 0.989 indicates that this model is the best-fitted model among the three.

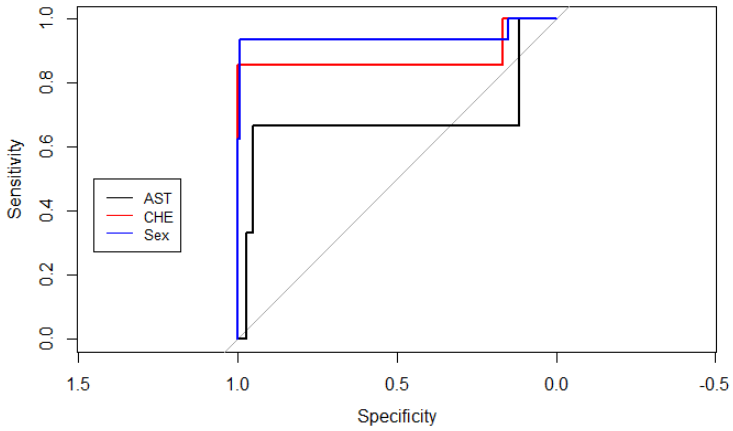


Figure 10: ROC plot for 3 random effects of GLMM model on test set

	threshold	specificity	sensitivity	accuracy	precision
AST	8.117251e-06	0.952381	0.6666667	0.9444444	0.2857143
CHE	0.3358878	1	0.8571429	0.9863014	1
Sex	0.4924259	0.9937888	0.9375	0.9887006	0.9375

Table 3: Evaluations for binary classification using 3 random effects of GLMM model

4.4 Neural Network

Lastly, we choose to build a Neural Network using Keras. The model is trained for 500 epochs. And the specificity and sensitivity are close to the one fitted in GLMM. Furthermore, we want to determine an appropriate number of epochs for this dataset, that is, how long do we need to train our model before it stops making any more progress.

	threshold	specificity	sensitivity	accuracy	precision
Neural Network	0.2198469	0.9689441	1	0.9491525	0.64

Table 4: Evaluations for binary classification using 3 random effects of GLMM model

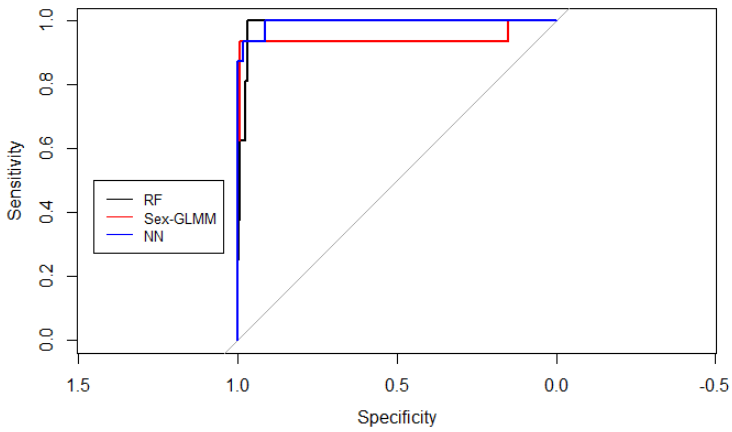


Figure 11: ROC plot for Random forest, best GLMM and Neural Network model on test set

From the plot, we can tell that the neural network model shows little improvement after 50 epochs, which costs quite a short time to fit the model. By setting the epochs to be 50±10, the callback of early stopping is a useful technique to prevent overfitting.

In binary classification, we mainly use sensitivity and specificity to measure the performance of the models. The conclusion from the ROC plot of Figure 11 is that the GLMM using Sex as a random effect outperforms the other two models in the aspect of accuracy.

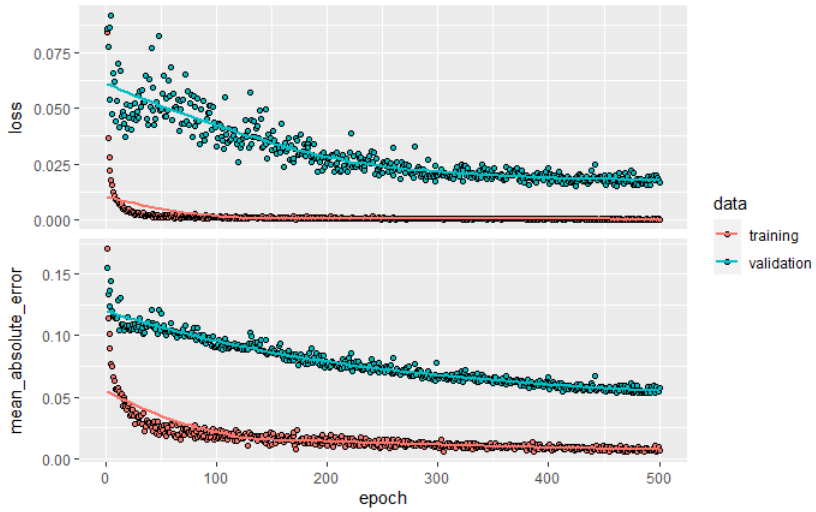


Figure 12: mae and loss against number of epoch (time)

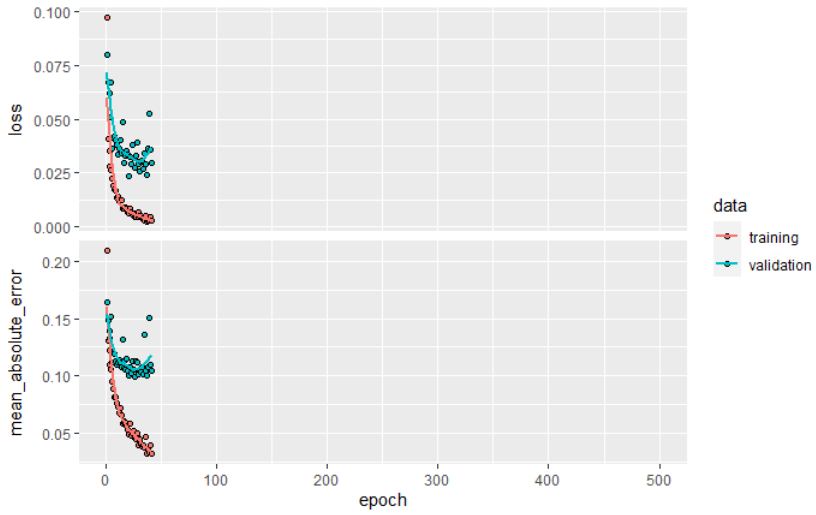


Figure 13: mae and loss against number of epoch (time)

5 Conclusion/Discussion

Initially, the neural network algorithm has an accuracy of 94.91% and the random forest algorithm provides us with the better performance that the accuracy is 97.18%. However, we were not satisfied with just simply fitting three models, the Gini importance feature of the random forest algorithm is then applied to select the most and least essential features.

From the analysis, it is found that AST, ALT and ALP are the three significant parameters for determining the presence of Hepatitis C. An important observation is that a better model is produced when using significant parameters that have a lower node purity as the random effect. For example, AST has a node purity of 2.982, whereas ALP has a node purity of 1.208, using ALP as the random effect would produce a better model.

Ultimately, from the result of our analysis, we would conclude that GLMM with the Sex parameter as the random effect works best in our situation as it gives a 98.87% accuracy - almost perfect. In further studies, it might be a worthy approach using the random forest algorithm to find the least important feature and apply it to the GLMM model.

From a medical point of view, when it comes to an initial screening for the presence of Hepatitis C in patients, physicians should pay close attention to the AST, ALT and ALP levels of the patients, as they are the best indicators for determining whether or not a patient has contracted Hepatitis C.

Moving forward, it is important to have patients maintain good levels of AST, ALT and ALP. In order to maintain a healthy level of AST, one can engage in regular exercise and increasing folic acid intake, such as legumes and eggs. Furthermore, to maintain a good level of ALT, one could increase their fiber intake, such as consuming additional fruits, as well as reducing the consumption of processed foods and saturated fats. Lastly, in order to maintain a proper level of ALP, it is recommended that people increase their consumption of vitamin B12, vitamin A, and healthy fats, which are commonly found in avocados, coconuts, yogurt, and fish.

6 Contributions

For the Contributions of the project, the Introduction and the Dataset description was written by Jeffrey. The Methods, description of the methods used both in the exploratory and main data analyses, was written by Jinhao. The Exploratory Data Analysis (EDA) was performed by Nicolaus. The Main Data Analysis was done by Rui and modified by Jinhao. Finally, the Conclusion/Discussion was done by Jeffrey, Jinhao and Nicolaus together.

References

1. World Health Organization. *Hepatitis C* https://www.who.int/news-room/fact-sheets/detail/hepatitis-c.
2. McCullagh, P. & Nelder, J. *Generalized Linear Models, Second Edition* ISBN: 9780412317606. http://books.google.com/books?id=h9kFH2%5C_FfBkC (Chapman & Hall, 1989).
3. Breslow, N. E. & Clayton, D. G. Approximate Inference in Generalized Linear Mixed Models. *Journal of the American Statistical Association* **88,** 9–25. ISSN: 01621459. http://www.jstor.org/stable/2290687 (1993).
4. UCLA: Statistical Consulting Group. *Introduction to Generalized Linear Mixed Models* https://stats.idre.ucla.edu/other/mult-pkg/introduction-to-generalized-linear-mixed-models/.
5. Ho, T. K. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20,** 832–844 (1998).
6. James, G., Witten, D., Hastie, T. & Tibshirani, R. *An Introduction to Statistical Learning: with Applications in R* https://faculty.marshall.usc.edu/gareth-james/ISL/ (Springer, 2013).
7. Menze, B. *et al.* A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data. *BMC Bioinformatics* **10,** 2–3 (2009).
8. Hastie, T., Tibshirani, R. & Friedman, J. *The Elements of Statistical Learning* (Springer New York Inc., New York, NY, USA, 2001).
9. Loecher, M. Unbiased variable importance for random forests. *Communications in Statistics - Theory and Methods* **0,** 1–13. eprint: https://doi.org/10.1080/03610926.2020.1764042. https://doi.org/10.1080/03610926.2020.1764042 (2020).

7. Appendix

Section 1. The Following Code Corresponds to Section 3, Exploratory Data Analysis – Code and Plots Referenced

```
suppressMessages(library(dbplyr))
#Load data
df = read.csv(file = 'hcvdat0.csv')
#Quick Look at the Data
print("Quick peek at the Data")
```

```
## [1] "Quick peek at the Data"
```

```
print(str(df))
```

```
## 'data.frame':    615 obs. of  14 variables:
##  $ X       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Category: chr  "0=Blood Donor" "0=Blood Donor" "0=Blood Donor" "0=Blood Dono
r" ...
##  $ Age     : int  32 32 32 32 32 32 32 32 32 32 ...
##  $ Sex     : chr  "m" "m" "m" "m" ...
##  $ ALB     : num  38.5 38.5 46.9 43.2 39.2 41.6 46.3 42.2 50.9 42.4 ...
##  $ ALP     : num  52.5 70.3 74.7 52 74.1 43.3 41.3 41.9 65.5 86.3 ...
##  $ ALT     : num  7.7 18 36.2 30.6 32.6 18.5 17.5 35.8 23.2 20.3 ...
##  $ AST     : num  22.1 24.7 52.6 22.6 24.8 19.7 17.8 31.1 21.2 20 ...
##  $ BIL     : num  7.5 3.9 6.1 18.9 9.6 12.3 8.5 16.1 6.9 35.2 ...
##  $ CHE     : num  6.93 11.17 8.84 7.33 9.15 ...
##  $ CHOL    : num  3.23 4.8 5.2 4.74 4.32 6.05 4.79 4.6 4.1 4.45 ...
##  $ CREA    : num  106 74 86 80 76 111 70 109 83 81 ...
##  $ GGT     : num  12.1 15.6 33.2 33.8 29.9 91 16.9 21.5 13.7 15.9 ...
##  $ PROT    : num  69 76.5 79.3 75.7 68.7 74 74.5 67.1 71.3 69.9 ...
## NULL
```

```
#Showing Interquartile range of each feature to understand their distribution
summary(df)
```

```
##        X           Category              Age             Sex
##  Min.   :  1.0   Length:615         Min.   :19.00   Length:615
##  1st Qu.:154.5   Class :character   1st Qu.:39.00   Class :character
##  Median :308.0   Mode  :character   Median :47.00   Mode  :character
##  Mean   :308.0                      Mean   :47.41
##  3rd Qu.:461.5                      3rd Qu.:54.00
##  Max.   :615.0                      Max.   :77.00
##
##       ALB             ALP              ALT              AST
##  Min.   :14.90   Min.   : 11.30   Min.   :  0.90   Min.   : 10.60
##  1st Qu.:38.80   1st Qu.: 52.50   1st Qu.: 16.40   1st Qu.: 21.60
##  Median :41.95   Median : 66.20   Median : 23.00   Median : 25.90
##  Mean   :41.62   Mean   : 68.28   Mean   : 28.45   Mean   : 34.79
##  3rd Qu.:45.20   3rd Qu.: 80.10   3rd Qu.: 33.08   3rd Qu.: 32.90
##  Max.   :82.20   Max.   :416.60   Max.   :325.30   Max.   :324.00
##  NA's   :1       NA's   :18       NA's   :1
##       BIL              CHE              CHOL            CREA
```

```
##  Min.   :  0.8   Min.   : 1.420   Min.   :1.430   Min.   :   8.00
##  1st Qu.:  5.3   1st Qu.: 6.935   1st Qu.:4.610   1st Qu.:  67.00
##  Median :  7.3   Median : 8.260   Median :5.300   Median :  77.00
##  Mean   : 11.4   Mean   : 8.197   Mean   :5.368   Mean   :  81.29
##  3rd Qu.: 11.2   3rd Qu.: 9.590   3rd Qu.:6.060   3rd Qu.:  88.00
##  Max.   :254.0   Max.   :16.410   Max.   :9.670   Max.   :1079.10
##                                   NA's   :10
##       GGT             PROT
##  Min.   :  4.50   Min.   :44.80
##  1st Qu.: 15.70   1st Qu.:69.30
##  Median : 23.30   Median :72.20
##  Mean   : 39.53   Mean   :72.04
##  3rd Qu.: 40.20   3rd Qu.:75.40
##  Max.   :650.90   Max.   :90.00
##                   NA's   :1
```

```
#Identify Number of Features in the data
#Number of Columns/Features in the Data
print(c("Num Cols/Features: ", ncol(df)))
```

```
## [1] "Num Cols/Features: " "14"
```

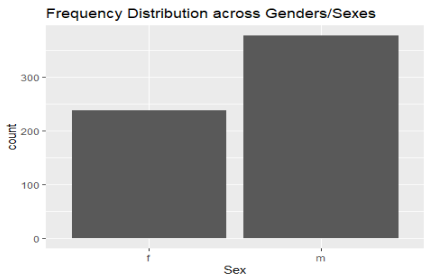
```
#Number of Rows in the Data
print(c("Num of Observations: ", nrow(df)))
```

```
## [1] "Num of Observations: " "615"
```

```
#Count Number of NAs per column
sapply(df, function(x) sum(is.na(x)))
```

```
##        X Category      Age      Sex      ALB      ALP      ALT      AST
##        0        0        0        0        1       18        1        0
##      BIL      CHE     CHOL     CREA      GGT     PROT
##        0        0       10        0        0        1
```

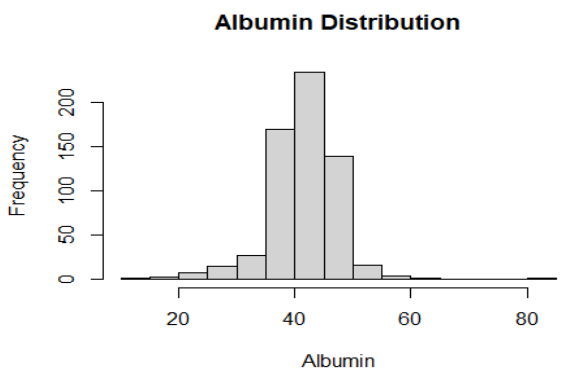
```
suppressMessages(library(dbplyr))
suppressMessages(library(ggplot2))

#histogram for age
hist(df$Age, main = "Histogram of Age Distribution", xlab = "Age")

#histogram to see categories
ggplot(data = df) + geom_bar(mapping = aes(x = Category)) + ggtitle("Frequency Dis
tribution across Categories")

#histogram for sex
ggplot(data = df) + geom_bar(mapping = aes(x = Sex)) + ggtitle("Frequency Distribu
tion across Genders/Sexes")
```

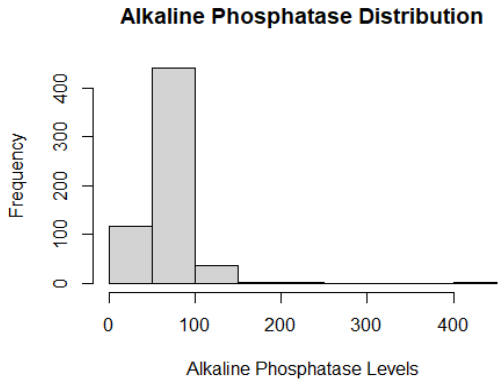


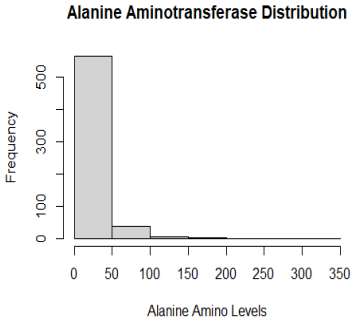

```r
library(janitor)

#breaking down the Categories by sex to understand the division
tabyl(df, Sex, Category)
```

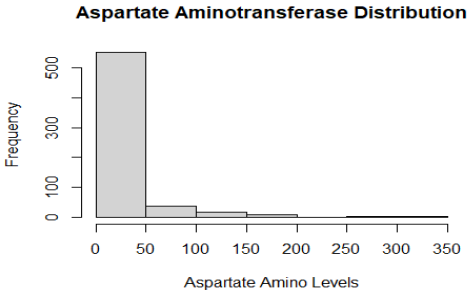
```
##  Sex 0=Blood Donor 0s=suspect Blood Donor 1=Hepatitis 2=Fibrosis 3=Cirrhosis
##    f           215                     1           4          8          10
##    m           318                     6          20         13          20
```

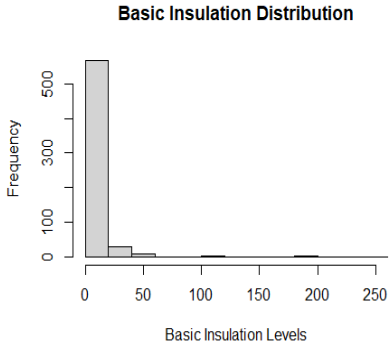
```r
#Doing a histogram for each feature to understand their distribution
hist(df$ALB, main = "Albumin Distribution", xlab = "Albumin")

hist(df$ALP, main = "Alkaline Phosphatase Distribution", xlab = "Alkaline Phosphat
ase Levels")

hist(df$ALT, main = "Alanine Aminotransferase Distribution", xlab = "Alanine Amino
Levels")

hist(df$AST, main = "Aspartate Aminotransferase Distribution", xlab = "Aspartate A
mino Levels")

hist(df$BIL, main = "Basic Insulation Distribution", xlab = "Basic Insulation Leve
ls")

hist(df$CHE, main = "Cholinesterase Levels Distribution", xlab = "Cholinesterase L
evels")

hist(df$CHOL, main = "Cholesteral Levels Distribution", xlab = "Cholesteral Levels
")

hist(df$CREA, main = "Creatinine Levels Distribution", xlab = "Creatinine Levels")

hist(df$GGT, main = "Gamma-Glutamyl Transferase Levels Distribution", xlab = "Gamm
a-Glutamyl Transferase Levels")

hist(df$PROT, main = "Protein Levels Distriibution", xlab = "Protein Levels")

#Printing Boxplots, to understand the Categories and how they are across the ages
ggplot(data = df) + geom_boxplot(mapping = aes(reorder(Category, Age, FUN = median
), y = Age)) + coord_flip() + labs(title = "Box-Plot of Categories across Ages", x
= "Category")

#Printing Boxplots to understand the gender distribution across the ages
ggplot(data = df) + geom_boxplot(mapping = aes(reorder(Sex, Age, FUN = median), y
= Age)) + coord_flip() + ggtitle("Box-Plot of Gender across Ages") + xlab("Gender"
)
```

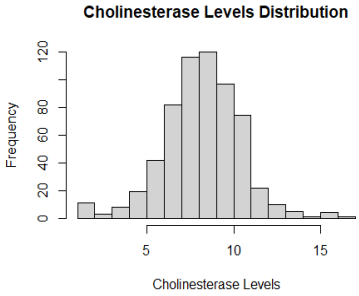


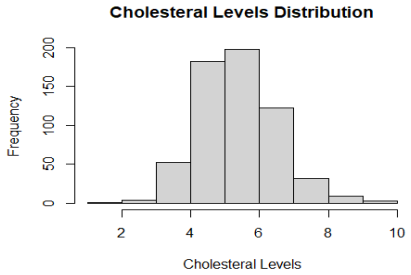


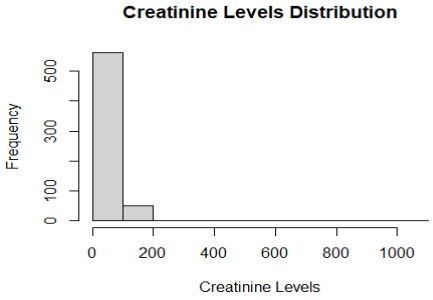


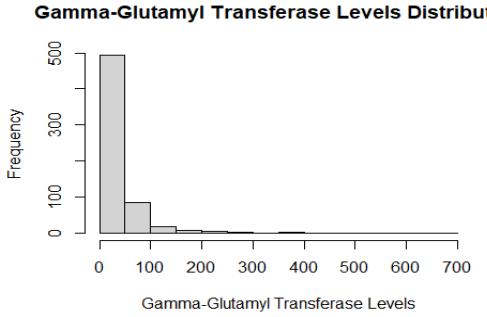


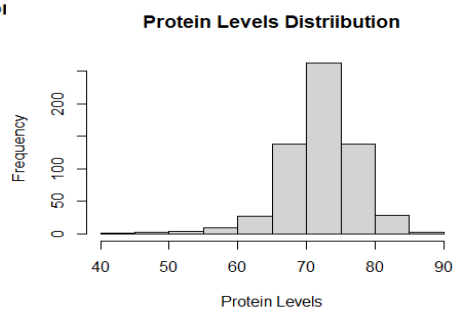












Explanatory analysis

```
HCVdata <- read.csv(file = 'hcvdat0.csv')
suppressMessages(library(dplyr))
drop <- c("X")
```

```
HCVdata = HCVdata[,!(names(HCVdata) %in% drop)]
colnames(HCVdata) = c("Category","Age","Sex","ALB","ALP","ALT","AST",
                      "BIL","CHE","CHOL","CREA","GGT","PROT")

# 1 and 2 for blood donor and suspect blood donor, 2,3,4 for hepatitis, fibrosis,c
irrhosis respectively
HCVdata$Category <- factor(HCVdata$Category,labels=c(1,2,3,4,5),
                           levels=c('0=Blood Donor',
                                    '0s=suspect Blood Donor',
                                    '1=Hepatitis',
                                    '2=Fibrosis',
                                    '3=Cirrhosis'))



# 0 for male, 1 for female
HCVdata$Sex <- factor(HCVdata$Sex,levels=c("m","f"),labels=c("0","1"))
HCVdata=na.omit(HCVdata[c("Category","Age","Sex","ALB","ALP","ALT","AST",
                          "BIL","CHE","CHOL","CREA","GGT","PROT")])
for(var in 1:13)
{
  HCVdata[,var]=as.numeric(HCVdata[,var])
}

tail(HCVdata)
```

```
##     Category Age Sex ALB   ALP  ALT   AST BIL  CHE CHOL  CREA   GGT PROT
## 608        5  52   2  39  37.0  1.3  30.4  21 6.33 3.78 158.2 142.5 82.7
## 609        5  58   2  34  46.4 15.0 150.0   8 6.26 3.98  56.0  49.7 80.6
## 610        5  59   2  39  51.3 19.6 285.8  40 5.77 4.51 136.1 101.1 70.5
## 611        5  62   2  32 416.6  5.9 110.3  50 5.57 6.30  55.7 650.9 68.5
## 612        5  64   2  24 102.8  2.9  44.4  20 1.54 3.02  63.0  35.9 71.3
## 613        5  64   2  29  87.3  3.5  99.0  48 1.66 3.63  66.7  64.2 82.0
```

EDA!!!

```
#class information, distribution of "category"
library(dplyr)
HCVdata %>%
  group_by(Category) %>%
  summarise(no_rows = length(Category))
```

```
## # A tibble: 5 x 2
##   Category no_rows
##      <dbl>   <int>
## 1        1     526
## 2        2       7
## 3        3      20
## 4        4      12
## 5        5      24
```

```
#mean of each variable
colMeans(HCVdata)
```

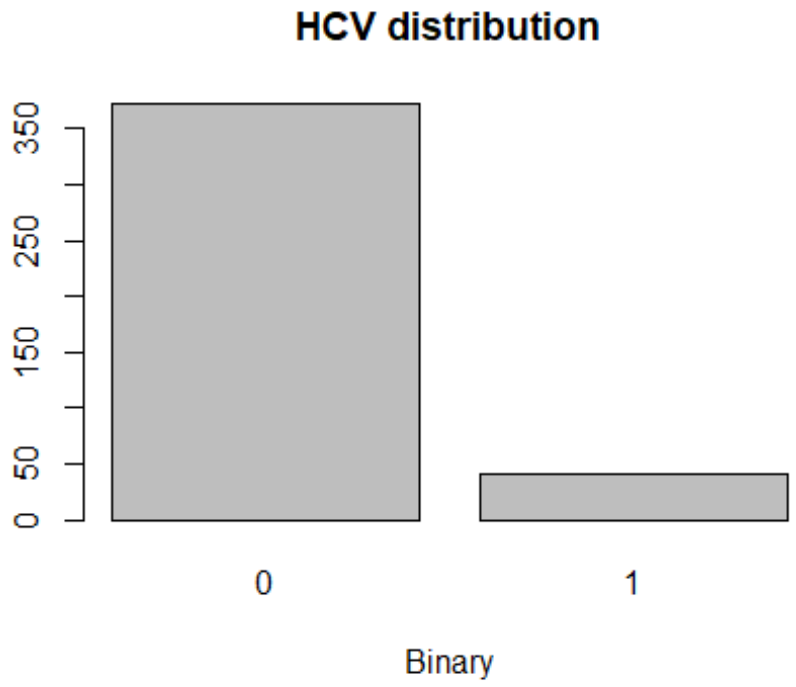
```
##  Category       Age       Sex       ALB       ALP       ALT       AST       BIL
##  1.303905 47.417657  1.383701 41.624278 68.123090 26.575382 33.772835 11.018166
##       CHE      CHOL      CREA       GGT      PROT
##  8.203633  5.391341 81.669100 38.198472 71.890153

#median of each variable
apply(HCVdata,2,median)

## Category      Age      Sex      ALB      ALP      ALT      AST      BIL
##     1.00    47.00     1.00    41.90    66.20    22.70    25.70     7.10
##      CHE     CHOL     CREA      GGT     PROT
##     8.26     5.31    77.00    22.80    72.10
```

train and test split 7:3

```
## train and test set 7:3
HCVdata<-HCVdata[complete.cases(HCVdata),]
set.seed(4850)
sample <- sample.int(n = nrow(HCVdata), size = floor(.70*nrow(HCVdata)), replace =
F)
train <- HCVdata[sample, ]
test <- HCVdata[-sample, ]
```

roc with random forest

```
#binary category
#1 for HCV, 0 for donor(regular/suspect)
set.seed(4850)
train$binary <- ifelse(as.numeric(train$Category)>2,1,0)
test$binary <- ifelse(as.numeric(test$Category)>2,1,0)
#barplot on binary
counts <- table(train$binary)
barplot(counts, main="HCV distribution",
   xlab="Binary")
```




```
#Load libraries
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

rfmod <- randomForest(formula = binary ~ .-Category, data = train, ntree = 10, max
nodes= 100, norm.votes = F)

## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?

suppressMessages(library(dplyr))
testm <- mutate(test,
                rf_pred = predict(rfmod,newdata = test,
                                  type="response"))

suppressMessages(library(pROC))

rocrf <- roc(testm$binary,testm$rf_pred)

## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases

AUCrf <- auc(rocrf)
rs <- rocrf[['rocs']]
```

Mixed model GLMM

binary random: AST

```
suppressMessages(library(ggpubr))
library(ggplot2)
agePlot=ggplot(train, aes(x=Age,y=binary))+geom_point()
ALPPlot=ggplot(train, aes(x=ALP,y=binary))+geom_point()
ALBPlot=ggplot(train, aes(x=ALB,y=binary))+geom_point()

# regular logistic model
bmod <- glm(binary ~.-Category,binomial,data=,train)

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

#GLMM model random effect:Age
library(MASS)

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

set.seed(4850)
suppressMessages(modpql1 <- glmmPQL(binary ~Sex+ALB+Age+ALT+ALT+BIL+CHE+CHOL+CREA+
GGT+PROT,
                  data=train, random= ~ 1|AST, family=binomial))

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

suppressMessages(drop1(modpql1, test="Chi"))

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Single term deletions
##
## Model:
## binary ~ Sex + ALB + Age + ALT + ALT + BIL + CHE + CHOL + CREA +
##     GGT + PROT
##        Df AIC LRT Pr(>Chi)
## <none>
## Sex     1
## ALB     1
## Age     1
## ALT     1
## BIL     1
## CHE     1
## CHOL    1
## CREA    1
## GGT     1
## PROT    1

testm1 <- mutate(test,
                 glmm_pred = predict(modpql1,newdata = test,
                                     type="response"))

suppressMessages(library(pROC))

roc1 <- roc(testm1$binary,testm1$glmm_pred)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

AUC <- auc(roc1)
rs <- roc1[['rocs']]
coords(roc1,"best",ret=c("threshold",
"specificity", "sensitivity", "accuracy","precision"))

##               threshold specificity sensitivity  accuracy precision
## threshold 8.117251e-06    0.952381   0.6666667 0.9444444 0.2857143
```

Random effect:CHE

```
#GLMM model random effect:CHE
library(MASS)
set.seed(4850)
suppressMessages(modpql2 <- glmmPQL(binary ~Sex+ALB+Age+ALT+ALT+BIL+AST+CHOL+CREA+
GGT+PROT,
                  data=train, random= ~ 1|CHE, family=binomial))

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

suppressMessages(drop1(modpql2, test="Chi"))

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Single term deletions
##
## Model:
## binary ~ Sex + ALB + Age + ALT + ALT + BIL + AST + CHOL + CREA +
##     GGT + PROT
##        Df AIC LRT Pr(>Chi)
## <none>
## Sex     1
## ALB     1
## Age     1
## ALT     1
## BIL     1
## AST     1
## CHOL    1
## CREA    1
## GGT     1
## PROT    1

testm2 <- mutate(test,
                 glmm_pred = predict(modpql2,newdata = test,
                                     type="response"))

suppressMessages(library(pROC))

roc2 <- roc(testm2$binary,testm2$glmm_pred)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

coords(roc2,"best",ret=c("threshold",
"specificity", "sensitivity", "accuracy","precision"))

##           threshold specificity sensitivity  accuracy precision
## threshold 0.3358878           1   0.8571429 0.9863014         1
```

Random effect:Sex

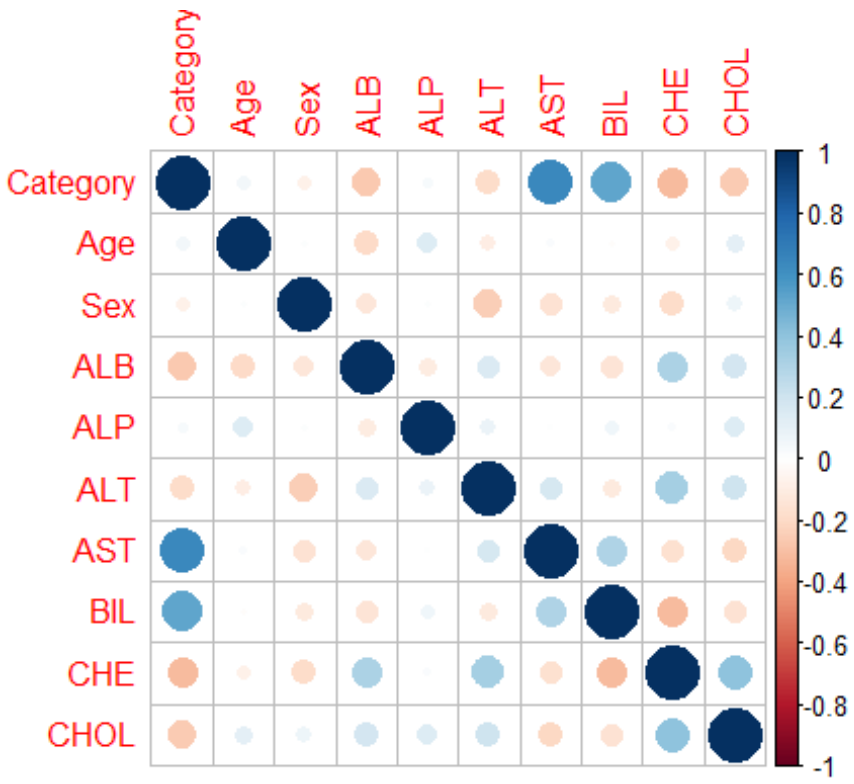
```
#GLMM model random effect:Sex
library(MASS)
set.seed(4850)
suppressMessages(modpql3 <- glmmPQL(binary ~CHE+ALB+Age+ALT+ALT+BIL+AST+CHOL+CREA+
GGT+PROT,
                  data=train, random= ~ 1|Sex, family=binomial))

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

suppressMessages(drop1(modpql3, test="Chi"))

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Single term deletions
##
## Model:
## binary ~ CHE + ALB + Age + ALT + ALT + BIL + AST + CHOL + CREA +
##     GGT + PROT
##        Df AIC LRT Pr(>Chi)
## <none>
## CHE     1
## ALB     1
## Age     1
## ALT     1
## BIL     1
## AST     1
## CHOL    1
## CREA    1
## GGT     1
## PROT    1

testm3 <- mutate(test,
                 glmm_pred = predict(modpql3,newdata = test,
                                     type="response"))
```

```r
suppressMessages(library(pROC))

roc3 <- roc(testm3$binary,testm3$glmm_pred)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

coords(roc3,"best",ret=c("threshold",
"specificity", "sensitivity", "accuracy","precision"))

##           threshold specificity sensitivity  accuracy precision
## threshold 0.4924259   0.9937888      0.9375 0.9887006    0.9375
```

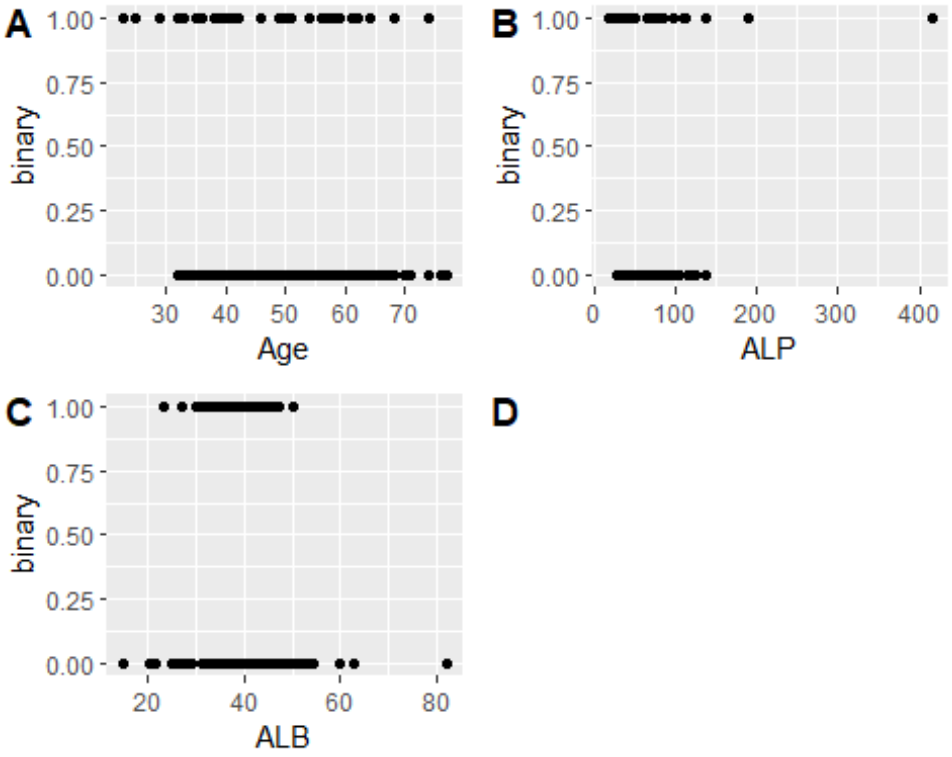
Deep learning

binary

```r
library(dplyr)
library(keras)
library(tensorflow)
library(tfdatasets)



set.seed(4850)
# Store the overall correlation in `M`
M <- cor(train[,1:10])

# Plot the correlation plot with `M`
library(corrplot)

## corrplot 0.84 loaded

predictorsCorr=corrplot(M, method="circle")
```




```
figure <- ggarrange(agePlot, ALPPlot, ALBPlot,predictorsCorr,
                    labels = c("A", "B", "C","D"),
                    ncol = 2, nrow = 2)

## Warning in as_grob.default(plot): Cannot convert object of class matrixarray
## into a grob.

figure
```




```
#normalize
spec <- feature_spec(train, binary ~ .-Category ) %>%
  step_numeric_column(all_numeric(), normalizer_fn = scaler_standard()) %>%
  fit()

## Warning in normalizePath(path.expand(path), winslash, mustWork): path[1]="C:
## \Users\Admin\.conda\envs\test-env/python.exe": The system cannot find the file
## specified

## Warning in normalizePath(path.expand(path), winslash, mustWork): path[1]="C:
## \Users\Admin\.conda\envs\test-env/python.exe": The system cannot find the file
## specified

spec

## -- Feature Spec ----------------------------------------------------------
-
## A feature_spec with 12 steps.
## Fitted: TRUE
## -- Steps -----------------------------------------------------------------
-
## The feature_spec has 1 dense features.
## StepNumericColumn: Age, Sex, ALB, ALP, ALT, AST, BIL, CHE, CHOL, CREA, GGT, PRO
T
## -- Dense features --------------------------------------------------------
-

layer <- layer_dense_features(
  feature_columns = dense_features(spec),
  dtype = tf$float32
```

```
)
suppressMessages(layer(train))
```

```
## tf.Tensor(
## [[ 0.653347   -0.6670947  -0.568868   ... -0.4312038   0.7929118
##   -0.76765865]
##  [-0.44780117 -0.22397378 -0.13247223 ... -0.32766175 -0.35917082
##   -0.76765865]
##  [-0.05207561  2.561358    0.41447717 ...  1.2864846   0.079717
##    1.2995006 ]
##  ...
##  [-0.9811694  -0.68571323 -1.2147336  ...  0.36570016  0.55518115
##   -0.76765865]
##  [-1.1188128   0.05530416 -0.1441094  ... -0.14276527 -2.059866
##    1.2995006 ]
##  [ 0.08556774 -0.71922666  0.8683287  ...  0.6338001  -0.03000497
##   -0.76765865]], shape=(412, 12), dtype=float32)
```

```
input <- layer_input_from_dataset(train[,2:13])


output <- input %>%
  layer_dense_features(dense_features(spec)) %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 1)

dpmod <- keras_model(input, output)
```

```
dpmod %>%
  compile(
    loss = "mse",
    optimizer = optimizer_rmsprop(),
    metrics = list("mean_absolute_error")
  )

build_model <- function() {
  input <- layer_input_from_dataset(train[,2:13])

  output <- input %>%
    layer_dense_features(dense_features(spec)) %>%
    layer_dense(units = 64, activation = "relu") %>%
    layer_dense(units = 64, activation = "relu") %>%
    layer_dense(units = 1)

  dpmod <- keras_model(input, output)

  dpmod %>%
```

```
    compile(
      loss = "mse",
      optimizer = optimizer_rmsprop(),
      metrics = list("mean_absolute_error")
    )

  dpmod
}


# Display training progress by printing a single dot for each completed epoch.
print_dot_callback <- callback_lambda(
  on_epoch_end = function(epoch, logs) {
    if (epoch %% 80 == 0) cat("\n")
    cat(".")
  }
)

dpmod <- build_model()

history <- dpmod %>% fit(
  x = train[2:13],
  y = train$binary,
  epochs = 500,
  validation_split = 0.2,
  verbose = 0,
  callbacks = list(print_dot_callback)
)

##
## ....................................................................................
.
## ....................................................................................
.
## ....................................................................................
.
## ....................................................................................
.
## ....................................................................................
.
## ....................................................................................
.
## ....................

library(ggplot2)

plot(history)

## `geom_smooth()` using formula 'y ~ x'
```

```
# it will stop when no more improvement
early_stop <- callback_early_stopping(monitor = "val_loss", patience = 20)

dpmod <- build_model()

history <- dpmod %>% fit(
  x = train[2:13],
  y = train$binary,
  epochs = 500,
  validation_split = 0.2,
  verbose = 0,
  callbacks = list(early_stop)
)

plot(history)
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
suppressMessages(library(dplyr))
testm <- mutate(test,
                dp_pred = dpmod %>% predict(test[,2:13]), type="response")



rocnn <- roc(testm$binary,testm$dp_pred)
```

```
## Setting levels: control = 0, case = 1
```

```
## Warning in roc.default(testm$binary, testm$dp_pred): Deprecated use a matrix as
## predictor. Unexpected results may be produced, please pass a numeric vector.
```

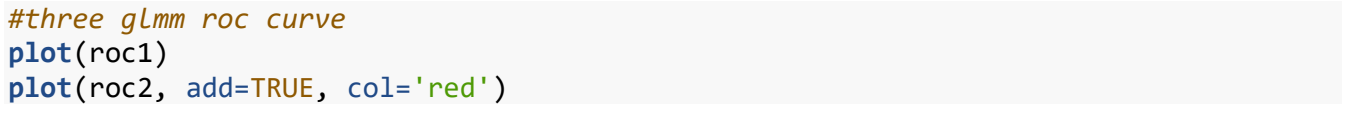
```
## Setting direction: controls < cases
```

```
AUCnn <- auc(rocnn)
rs <- rocnn[['rocs']]
suppressMessages(summary(dpmod))
```

```
## Model: "model_2"
## _________________________________________________________________________
_
## Layer (type)                 Output Shape              Param #   Connected to
## =========================================================================
=
## ALB (InputLayer)             [(None,)]                 0
## _________________________________________________________________________
_
## ALP (InputLayer)             [(None,)]                 0
## _________________________________________________________________________
_
## ALT (InputLayer)             [(None,)]                 0
## _________________________________________________________________________
```

```
_
## AST (InputLayer)          [(None,)]         0
## _____________________________________________________________________________
_
## Age (InputLayer)          [(None,)]         0
## _____________________________________________________________________________
_
## BIL (InputLayer)          [(None,)]         0
## _____________________________________________________________________________
_
## CHE (InputLayer)          [(None,)]         0
## _____________________________________________________________________________
_
## CHOL (InputLayer)         [(None,)]         0
## _____________________________________________________________________________
_
## CREA (InputLayer)         [(None,)]         0
## _____________________________________________________________________________
_
## GGT (InputLayer)          [(None,)]         0
## _____________________________________________________________________________
_
## PROT (InputLayer)         [(None,)]         0
## _____________________________________________________________________________
_
## Sex (InputLayer)          [(None,)]         0
## _____________________________________________________________________________
_
## dense_features_3 (DenseFe (None, 12)        0         ALB[0][0]
##                                                       ALP[0][0]
##                                                       ALT[0][0]
##                                                       AST[0][0]
##                                                       Age[0][0]
##                                                       BIL[0][0]
##                                                       CHE[0][0]
##                                                       CHOL[0][0]
##                                                       CREA[0][0]
##                                                       GGT[0][0]
##                                                       PROT[0][0]
##                                                       Sex[0][0]
## _____________________________________________________________________________
_
## dense_8 (Dense)           (None, 64)        832       dense_features_3[0][0]
## _____________________________________________________________________________
_
## dense_7 (Dense)           (None, 64)        4160      dense_8[0][0]
## _____________________________________________________________________________
_
## dense_6 (Dense)           (None, 1)         65        dense_7[0][0]
## =============================================================================
=
## Total params: 5,057
```

```
## Trainable params: 5,057
## Non-trainable params: 0
## _____________________________________________________________________________
_

#RM,(AST ALT ALP) glmm, nn
coords(rocrf,"best",ret=c("threshold",
"specificity", "sensitivity", "accuracy","precision"))##the random forest model

##           threshold specificity sensitivity  accuracy precision
## threshold 0.2833333   0.9689441           1 0.9717514 0.7619048

coords(roc1,"best",ret=c("threshold",
"specificity", "sensitivity", "accuracy","precision"))# the AST GLMM model

##              threshold specificity sensitivity  accuracy precision
## threshold 8.117251e-06    0.952381   0.6666667 0.9444444 0.2857143

coords(roc2,"best",ret=c("threshold",
"specificity", "sensitivity", "accuracy","precision"))#the CHE GLMM model

##           threshold specificity sensitivity  accuracy precision
## threshold 0.3358878           1   0.8571429 0.9863014         1

coords(roc3,"best",ret=c("threshold",
"specificity", "sensitivity", "accuracy","precision"))# the Sex GLMM model

##           threshold specificity sensitivity  accuracy precision
## threshold 0.4924259   0.9937888      0.9375 0.9887006    0.9375

coords(rocnn,"best",ret=c("threshold",
"specificity", "sensitivity", "accuracy","precision"))# the nn GLMM model

##           threshold specificity sensitivity  accuracy precision
## threshold 0.2867029   0.9937888      0.9375 0.9887006    0.9375

#importance: impurity gini importance
c(log(importance(rfmod)),importance(rfmod))

##  [1] -1.018644e+00 -4.042568e+01 -8.560265e-01  1.207540e+00  2.027953e+00
##  [6]  2.982227e+00  5.400411e-01  2.264169e-01 -1.000342e+00 -7.677341e-01
## [11]  9.831243e-01 -3.262736e+01  3.610843e-01  2.775558e-18  4.248468e-01
## [16]  3.345247e+00  7.598513e+00  1.973172e+01  1.716077e+00  1.254098e+00
## [21]  3.677537e-01  4.640634e-01  2.672794e+00  6.762646e-15

#neural network and random forest
plot(rocrf)
plot(rocnn, add=TRUE, col='blue')
legend(1.45, 0.5, legend=c("RF", "NN"),
       col=c( "black","blue"), lty=1, cex=0.8)
```



```
#three glmm roc curve
plot(roc1)
plot(roc2, add=TRUE, col='red')
```

```r
plot(roc3, add=TRUE, col='blue')
legend(1.45, 0.5, legend=c("AST", "CHE", "Sex"),
       col=c( "black","red", "blue"), lty=1, cex=0.8)
```

```r
#randomforest best glmm and nn roc curve
plot(rocrf)
plot(roc3, add=TRUE, col='red')
plot(rocnn, add=TRUE, col='blue')
legend(1.45, 0.5, legend=c("RF", "Sex-GLMM", "NN"),
       col=c( "black","red", "blue"), lty=1, cex=0.8)
```