Project VII: Artificial Neural Networks (ANN) and Suport Vector Machines (SVM)

DS 5494 - Statistical Data Mining II

Willliam Ofosu Agyapong*

December 10, 2022

Contents

*woagyapong@miners.utep.edu, University of Texas at El Paso (UTEP).

1 Introduction

In this project, we considered various predictive models including Regularized Logistic Regression, Random Forest, MARS, Artificial Neural Networks (ANN), and Support Vector Machines (SVM). The Random Forest and logistic regression models were included as baseline models for comparison. The **main goal of this project is to make medical diagnosis of Hepatitis C based on the results of lab blood work**. Specifically, we aim to obtain a model with the best predictive performance and also identify the most important predictors of of Hepatitis C incidence.

-

1.1 Data Description

We used the HCV data available at UCI machine learning repository: https://archive.ics.uci.edu/ml/datasets/HCV+data. Excluding the patient Id column, this 615×13 data set contains 10 laboratory values of blood donors and Hepatitis C patients and demographic values age and gender (sex). The target attribute for classification is `Category` (blood donors vs. Hepatitis C (including its progress ('just' Hepatitis C, Fibrosis, Cirrhosis). All attributes except Category and Sex are numerical.

2 Data Preparation

```
# SKIP FIRST COLUMN
dat  <- read.csv("hcvdat0.csv", header=TRUE, colClasses=c("NULL", rep(NA, 13)))
dim(dat);
```

```
## [1] 615  13
```

```
head(dat) %>%
    kable(booktabs=T, linesep="") %>%
    kable_styling(latex_options = c("HOLD_position"));
```

Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
0=Blood Donor	32	m	38.5	52.5	7.7	22.1	7.5	6.93	3.23	106	12.1	69.0
0=Blood Donor	32	m	38.5	70.3	18.0	24.7	3.9	11.17	4.80	74	15.6	76.5
0=Blood Donor	32	m	46.9	74.7	36.2	52.6	6.1	8.84	5.20	86	33.2	79.3
0=Blood Donor	32	m	43.2	52.0	30.6	22.6	18.9	7.33	4.74	80	33.8	75.7
0=Blood Donor	32	m	39.2	74.1	32.6	24.8	9.6	9.15	4.32	76	29.9	68.7
0=Blood Donor	32	m	41.6	43.3	18.5	19.7	12.3	9.92	6.05	111	91.0	74.0

```
anyNA(dat);
```

```
## [1] TRUE
```

```
# str(dat)
# table(dat$Category)
# unique(dat$Category)
# sum(is.na(dat$Category))
```

2.1 Part (a)

We modify the target `Category` into a binary response so that `Category = 0` if it falls into either "0=Blood Donor" or "0s=suspect Blood Donor" and 1 if it falls into any other category except being missing, in which case we keep it as is.

```
dat_new <- dat %>%
    mutate(Category = if_else(Category %in% c("0=Blood Donor",
```

```
                                                "0s=suspect Blood Donor"),
                                        0, 1))
table(dat_new$Category)
```

```
##
##   0   1
## 540  75
```

We see that there are 540 healthy blood donors and 75 Hepatitis C patients in the whole data set.

2.2 Part (b): Frequency Distribution of Target variable

```
# distribution table
last_row <- data.frame(Category="Total",
                       n=nrow(dat_new),
                       percent=100,
                       missing=sum(is.na(dat_new$Category)))

dat_new %>%
    mutate(Category = as.character(Category)) %>%
    group_by(Category) %>%
    summarise(n=n(), percent = n*100/nrow(dat_new)) %>%
    mutate(missing = c(NULL, NULL)) %>%
    bind_rows(last_row) %>%
    kable()

# distribution plot
dat_new %>%
  mutate(Category = factor(Category)) %>%
  group_by(Category) %>%
  summarise(n=n()) %>%
  mutate(pct = n/sum(n),
         lbl = scales::percent(pct)) %>%
ggplot(aes(Category, pct,fill=Category)) +
    geom_bar(stat = "identity",position = "dodge",alpha=0.7) +
    scale_fill_brewer(palette = "Set2") +
  geom_text(aes(label = lbl), size=3, color = "white",
            position = position_stack(vjust = 0.5)) +
   labs(y="Percent", title = "") +
    theme(legend.position = "none")
```

Category	n	percent	missing
0	540	87.8	NA
1	75	12.2	NA
Total	615	100.0	0

Table 1: Frequency distribution table

Table 2:

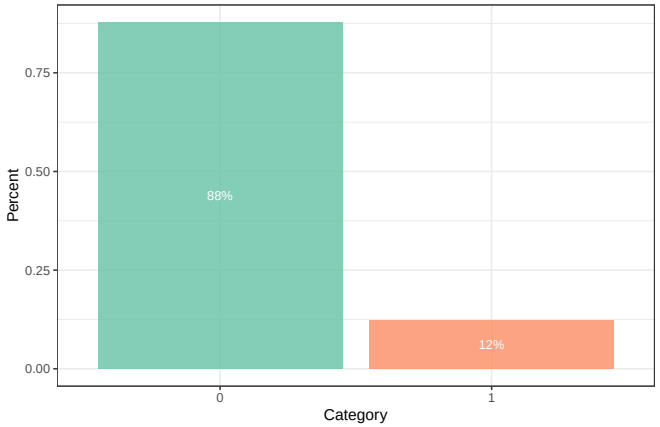


Figure 1: Distribution of Target

We observe from the above information that the number of observations for the **0** class exceeds that of the **1** class by 76%, signifying a very large difference in relation to the relatively small sample size of 615. Therefore, I consider this an instance of imbalanced classification problem, and as such, necessary measures such as applying a stratification strategy on the target variable when splitting the data into training and test sets will be considered in the predictive modeling section to deal with or lessen the impact of this potential problem.

It is also clear from Table 2 that there is no missing values in the target `Category`.

2.3 Part (c): Handling missing values in predictors

```
# get data types
output <- NULL
for(i in seq_along(dat_new)) {
  output <- rbind(output, c(names(dat_new)[i],
                  paste(class(dat_new[[i]]), collapse = " "),
                  length(unique(dat_new[[i]]))
                   )
                  )
}
output <- as.data.frame(output)

# checking for missing values
output %>% left_join(
naniar::miss_var_summary(dat_new), by=c("V1"="variable")) %>%
  filter(V1 !="Category") %>% # take out the target variable
  kable(booktabs = T, linesep="", align = "lcc",
        col.names = c("Variable name", "Type", "levels", "Number missing", "Percent missing"),
        cap = "Data types and amount of missing values in the predictors") %>%
kable_styling(latex_options = c("HOLD_position"))
```

Variable name	Type	levels	Number missing	Percent missing
Age	integer	49	0	0.0000
Sex	character	2	0	0.0000
ALB	numeric	190	1	0.1626
ALP	numeric	415	18	2.9268
ALT	numeric	342	1	0.1626
AST	numeric	297	0	0.0000
BIL	numeric	188	0	0.0000
CHE	numeric	407	0	0.0000
CHOL	numeric	314	10	1.6260
CREA	numeric	117	0	0.0000
GGT	numeric	358	0	0.0000
PROT	numeric	199	1	0.1626

Table 3: Data types and amount of missing values in the predictors

From Table 3, we see that five of the predictors, namely `ALB`, `ALP`, `ALT`, `CHOL` and `PROT`, have missing values, **with `ALP` dominating the list with 2.93% missing rate followed closely by `CHOL`**. Before proceeding, we will take a moment to impute these missing values using the `mice` package with Predictive Mean Matching (PMM) as the imputation method.

2.3.1 Missing value imputation with pmm method

```
# Missing value imputation for predictors
set.seed(123)
```

```
imputed_object <- mice::mice(dat_new[,-which(names(dat_new)=="Category")], m=5,
                             method = "pmm", maxit = 30, printFlag = F)
dat_imputed <- as.data.frame(complete(imputed_object, 1)) # use the first imputed data

# append the target variable
dat_imputed <- dat_imputed %>%
    mutate(Category = dat_new$Category, .before="Age")
rm(imputed_object)
# Confirm no missing values
naniar::miss_var_summary(dat_imputed) %>%
  kable(booktabs = T, linesep="", align = "lcc",
        col.names = c("Variable name", "Number missing", "Percent missing"),
        cap = "Status of data after missing values' imputation") %>%
kable_styling(latex_options = c("HOLD_position"))
```

Variable name	Number missing	Percent missing
Category	0	0
Age	0	0
Sex	0	0
ALB	0	0
ALP	0	0
ALT	0	0
AST	0	0
BIL	0	0
CHE	0	0
CHOL	0	0
CREA	0	0
GGT	0	0
PROT	0	0

Table 4: Status of data after missing values' imputation

The first predicted imputation was selected to fill the missing values in the dataset.* Of course we do not observe any missing values at this point after the imputation! Let's now head over to change the data matrix into numeric using the `model.matrix()` function. Here, dummy variables will be automatically created for each categorical predictor.

2.4 Part (d): Data matrix into numeric

```
dat_mat <- model.matrix(Category ~ .-1, data = dat_imputed)
dat_mat <- dat_mat[,-1]
model.dat <- data.frame(Category=dat_imputed$Category, dat_mat)
head(model.dat) %>%
  kable(booktabs = T, linesep="",
        cap = "First few observations from the resulting data matrix") %>%
kable_styling(latex_options = c("HOLD_position"))
```

Category	Sexf	Sexm	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
0	0	1	38.5	52.5	7.7	22.1	7.5	6.93	3.23	106	12.1	69.0
0	0	1	38.5	70.3	18.0	24.7	3.9	11.17	4.80	74	15.6	76.5
0	0	1	46.9	74.7	36.2	52.6	6.1	8.84	5.20	86	33.2	79.3
0	0	1	43.2	52.0	30.6	22.6	18.9	7.33	4.74	80	33.8	75.7
0	0	1	39.2	74.1	32.6	24.8	9.6	9.15	4.32	76	29.9	68.7
0	0	1	41.6	43.3	18.5	19.7	12.3	9.92	6.05	111	91.0	74.0

Table 5: First few observations from the resulting data matrix

3 Exploratory Data Analysis

3.1 Range and variations of the predictors

```
dat_imputed %>%
  tidyr::pivot_longer(-c(Category,Sex), names_to="Predictor", values_to="Value") %>%
  ggplot(aes(Predictor, Value, fill=Predictor)) +
  geom_boxplot() + ggtitle("Distribution of continuous predictors via parallel boxplots") +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 90, vjust = .5, hjust = 1))
```

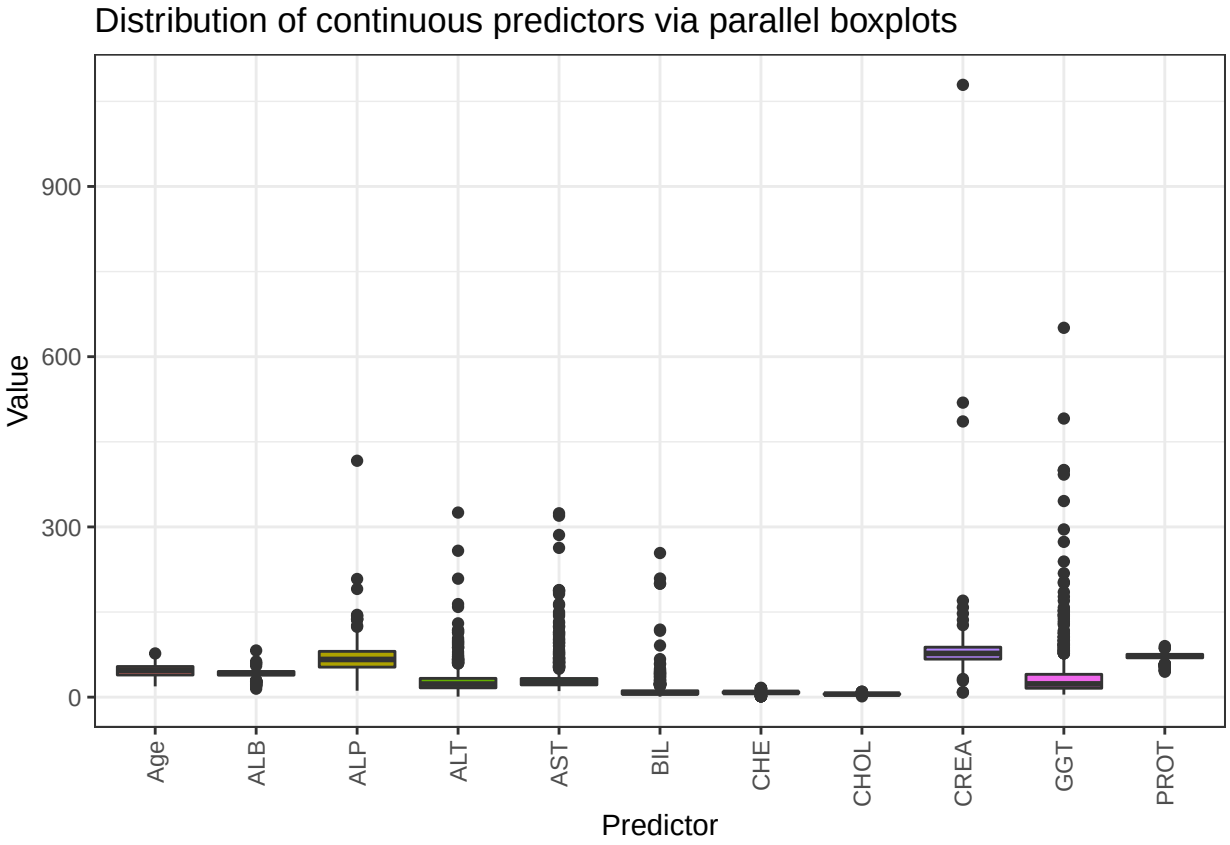


Figure 2: Parallel boxplots showing how the continuous predictors are distributed

Figure 2 depicts uneven range and variations across the predictors. *Some predictors have wide ranges and higher variations than others.* For instance, while `CHE` and `CHOL` exhibit similar distributions, their ranges and variations differ considerably from the rest of the predictors, especially `ALP`, `CREA`, and `GGT`. This calls for the need to normalize or standardize the predictors during the modeling phase, especially for some of the methods such as ANN.

3.2 Exploring associations between the target and predictors

3.2.1 Association between variables

```
# install.packages("GoodmanKruskal")
library(GoodmanKruskal)
cor_mat <- GKtauDataframe(dat_imputed)
plot(cor_mat, colorPlot=T)
```

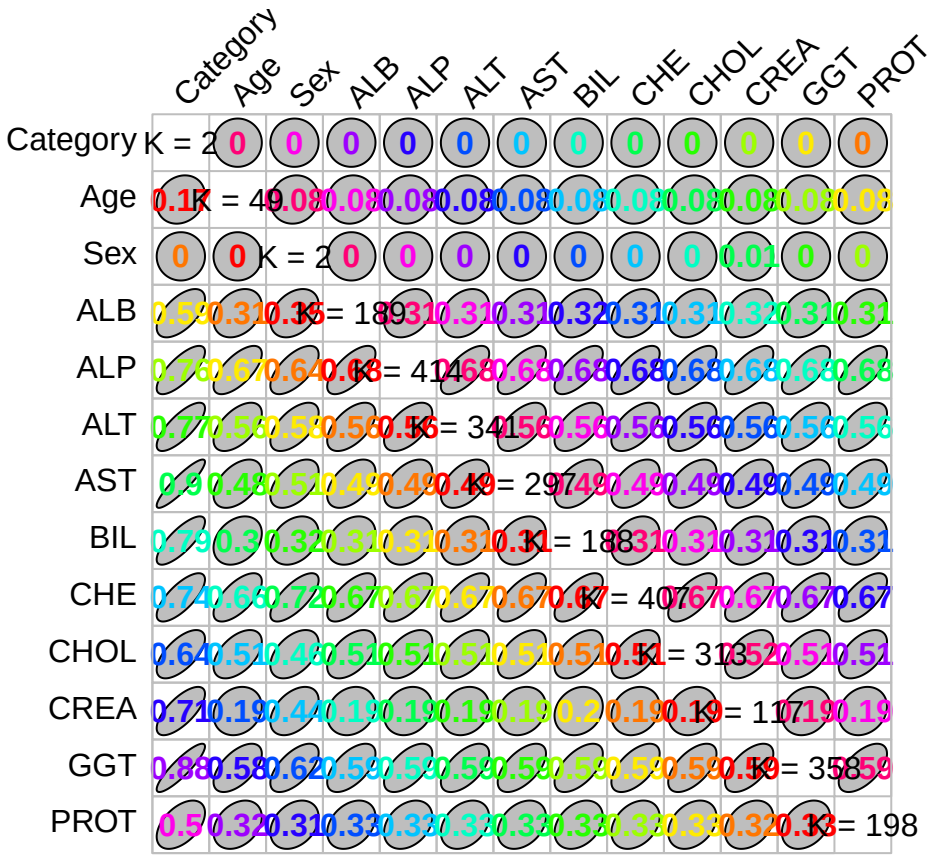


Figure 3: Association between variables

The above figure reveals that `AST`, `GGT`, and `BIL` appear highly predictive of Hepatitis C incidence, since there is a high association values of 0.9, 0.88 and 0.79, respectively, between each one of these predictors and the target `Category`. The other predictors are also moderately associated with `Category` except `Sex` recording a 0 association value with `Category`. Reading from the top columns to the right, we observe moderate associations among some of the variables. For instance, the association values between Age and ALP, Age and ALT, Age and CHE are 0.67, 0.56, 0.66, respectively. This may lead to a potential problem of multicollinearity.

3.2.2 Distribution of continuous predictors by `Category`

Alternatively, we also construct comparative box plots to assess the influence of the continuous predictors on the target variable.

```
dat_imputed %>%
    dplyr::select(-Sex) %>%
    mutate(Category = factor(Category)) %>%
    pivot_longer(-Category, names_to = "predictor", values_to = "value") %>%
    ggplot(aes(Category, value, fill = Category)) +
    geom_boxplot(alpha=0.8) +
    scale_fill_brewer(palette = "Set2") +
```

```
    facet_wrap(vars(predictor),  scales = "free") +
    labs(x="", y="") +theme_bw() +
    theme(axis.text.x = element_blank(),
          axis.ticks.x = element_blank())
```

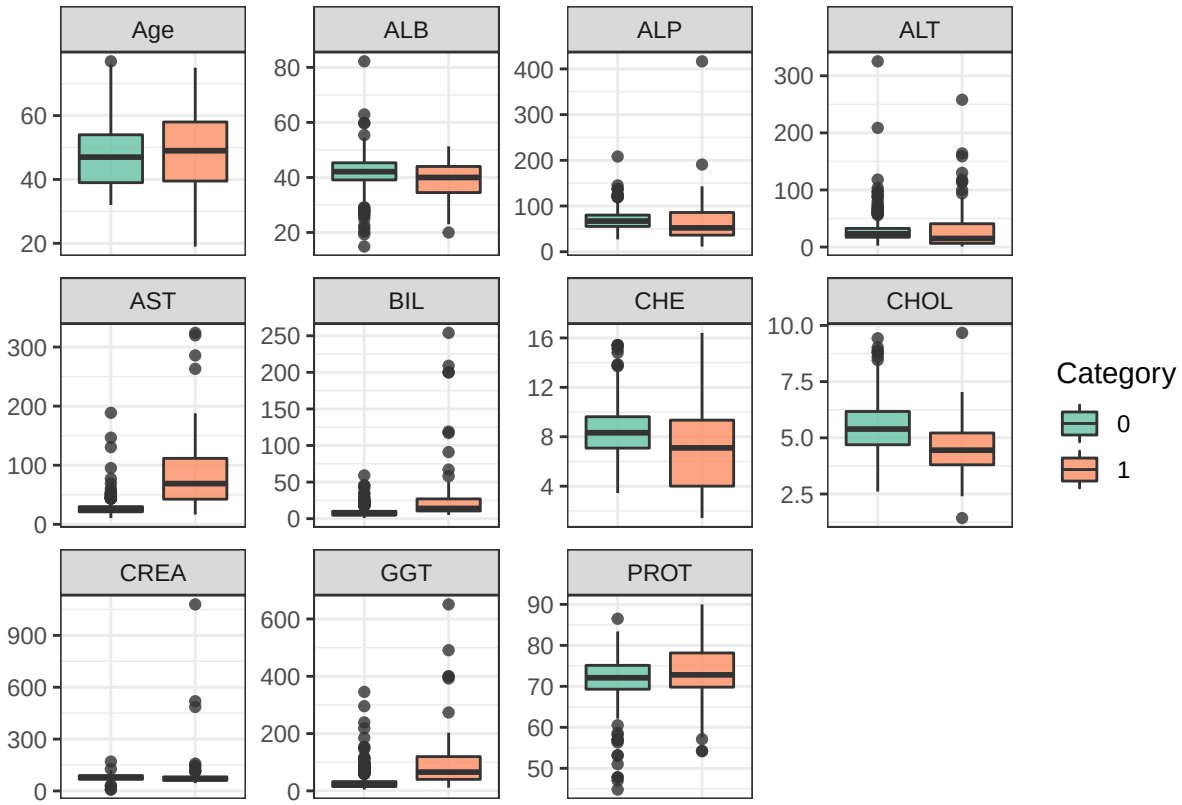


Figure 4: Distribution of continuous predictors by the target Category

Figure 4, we observe the following:

- The distributions of most of the features differ somewhat significantly across the two levels of the target `Category`, showing how influential they may be in predicting Hepatitis C incidence. We see this in the plots for AST, ALT, GGT, CHOL.
- Even though the highest age comes from the healthy blood donors, on average, Hepatitis C patients are older.
- Blood donors have higher ALB, ALP, CHE and CHOL values, and about the same BIL, ALT, and PROT than Hepatitis C patients on average.

3.2.3 Distribution of Sex by target variable

```
library(scales)
dat_imputed %>%
  mutate(Category = factor(Category),
         Sex = ifelse(Sex=="f", 'Female', 'Male')) %>%
  group_by(Sex, Category) %>%
  summarise(n=n()) %>%
  mutate(pct = n/sum(n),
         lbl = percent(pct)) %>%
  ggplot(aes(Sex, pct, fill=Category)) +
  geom_bar(stat = "identity",position = "fill",alpha=0.8) +
  scale_y_continuous(breaks = seq(0,1,0.2), label=percent) +
```

```
  geom_text(aes(label = lbl), size=3, position = position_stack(vjust = 0.5)) +
  scale_fill_brewer(palette="Set2") +
  labs(x="Sex",y="Percent of subjects", fill="Category") +
  theme_classic()
```

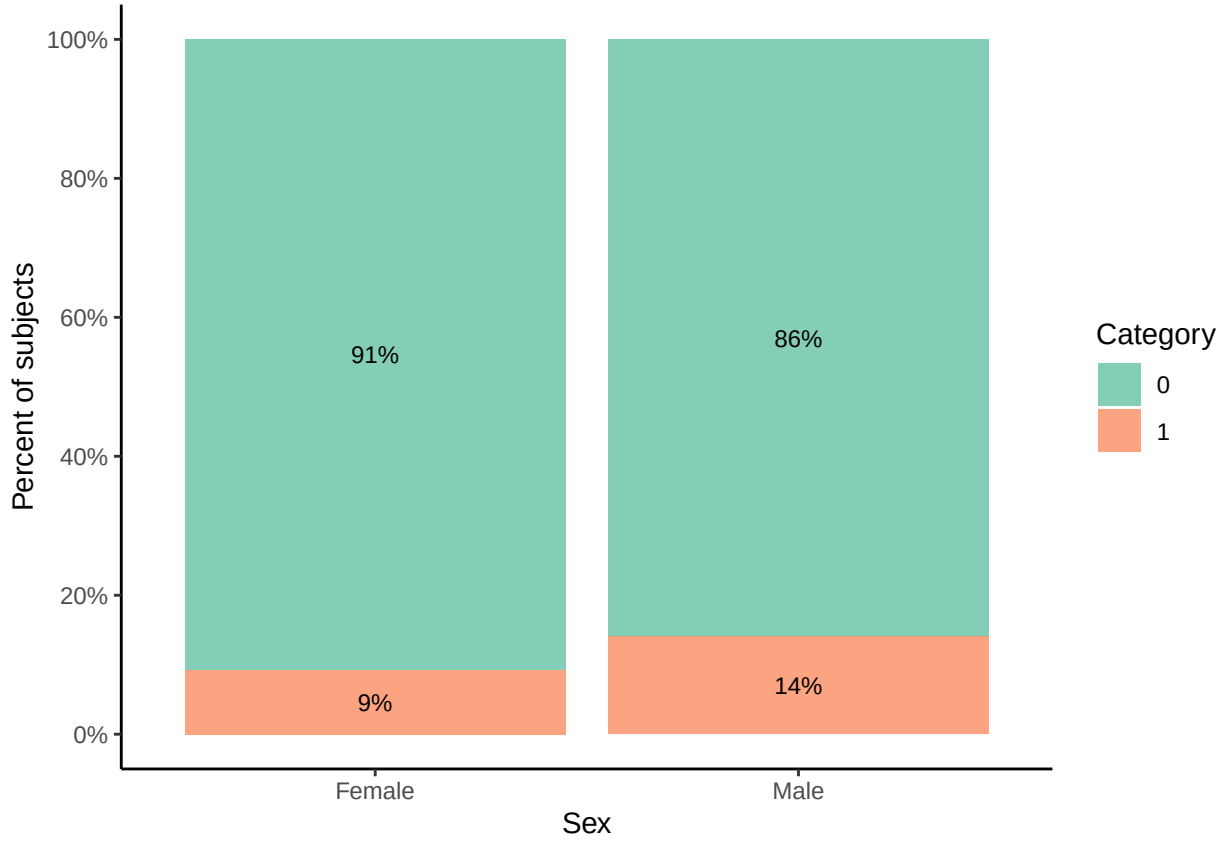


Figure 5: The effect of Sex on Hepatitis C incidence using a segmented bar plot

Contrary to what we observed from From Figure 3, Figure 5 provides us with some kind of association between the target `Category` and `Sex` as `Category` is not distributed equally for males and females. While there are more females in the 0 class than males, males are on the high side for the 1 class. To ascertain whether this difference is significant, we performed a chisquare test of independence.

Chi-square test for sex and category

```
with(dat_imputed, chisq.test(Category, Sex))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  Category and Sex
## X-squared = 2.7, df = 1, p-value = 0.1
```

From the test result, a p-value of 0.1 suggests that there is no statistically significance association between Sex and Category, which supports what we observed from the association graph. This means the differences observed in the above plots are not significant.

4 Outlier Detection

For this outlier detection problem, we used the one-class support vector machine anomaly detection technique. We first trained the model with data of healthy blood donors (`Category = 0`). Because this is an unsupervised

problem, only the predictors were used. We then apply the model to predict the data of Hepatitis C patients and see if the method is able to successfully distinguish most of the Hepatitis C patients from healthy donors. In other words, can the method detect Hepatitis C patients as outliers based on what is learned from healthy blood donors?

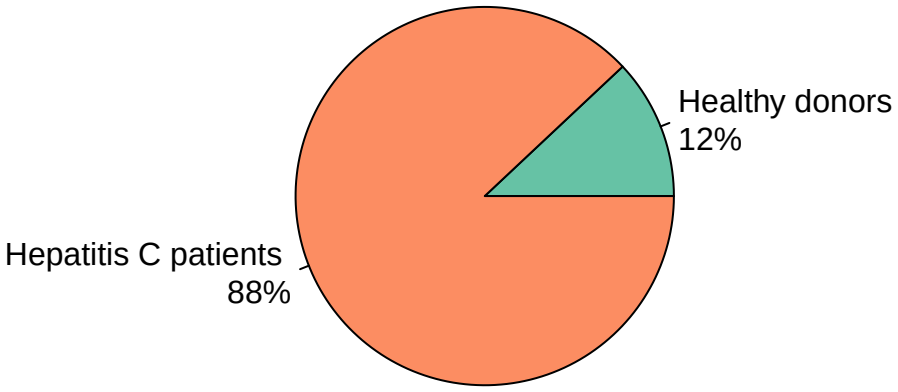
```
library(e1071)

# separate healthy blood donors from unhealthy (Hepatitis C) patients.
# dat_healthy <- dat_imputed %>%
#     filter(Category ==0) %>%
#     dplyr::select(-Category, -Sex)
#
# dat_HC <- dat_imputed %>%
#     filter(Category ==1) %>%
#     dplyr::select(-Category, -Sex)

dat_healthy <- model.dat %>%
    filter(Category ==0) %>%
    dplyr::select(-Category)

dat_HC <- model.dat %>%
    filter(Category ==1) %>%
    dplyr::select(-Category)


x <- dat_healthy
p <- NCOL(x)
fit.OneClassSVM <- svm(x, y=NULL, type="one-classification", nu=0.02, # nu - OC-SVM TUNING PARAMETER
    kernel="radial", gamma=1/p)  # gamma - PARAMETER IN RBF KERNEL
# summary(fit.OneClassSVM)

# test on the whole set

pred <- predict(fit.OneClassSVM, dat_HC)
table(pred) # show frequency distribution of the predictions
```

```
## pred
## FALSE  TRUE
##    66     9
```

```
# proportion of unhealthy donors detected as outliers
(prop.unhealthy <- (length(pred) - sum(pred)) / length(pred))
```

```
## [1] 0.88
```

```
prop.label <- paste0(c("Healthy donors \n","Hepatitis C patients \n"),
                     c((1-prop.unhealthy)*100, prop.unhealthy*100))
prop.label <- paste0(prop.label, "%")
pie(c("Healthy donors"=sum(pred), "Hepatitis C patients"=length(pred) - sum(pred)),
    labels = prop.label, col = c("#66C2A5", "#FC8D62"))
```



From the output, `TRUE` goes for the healthy blood donors on which the model was trained and `FALSE` indicates outliers (Hepatitis C patients in our case). with 88% proportion of unhealthy donors detected as outliers, it is evident that ***the one-class SVM method is able to successfully distinguish most of the Hepatitis C patients from the healthy donors***.

The model identified observation 19 as a potential outlier

5 Data Partitioning

To be able to validate our trained classifiers in Section 6 against an unseen data, instead of using the usual one time training-test split, we employed a V-fold cross-validation strategy with V=10 largely due to the fact that our sample size is relatively small. This strategy allows us to use the whole data for training the models and the whole data for predictions in a smart manner where one fold is set aside as the test set and the remaining 9 folds used as the training set while iterating over all 10 folds. Again, to ensure that similar proportions of 0s and 1s are preserved in each fold, we randomly split the data into 10 folds with stratification on the target variable `Category` as illustrated in the codes below. This is particularly necessary given that the 0s far outnumber the 1s as we observed in the EDA section. Table 6 shows the number of observations (samples) allocated to the training data and the test data for each fold.

```
set.seed(126)
# generate the 10-fold indexes
V <- 10
n <- NROW(dat_imputed); n0 <- sum(dat_imputed$Category==0); n1 <- n-n0;
id.fold <- 1:n
id.fold[dat_imputed$Category==0] <- sample(x=1:V, size=n0, replace=TRUE)
id.fold[dat_imputed$Category==1] <- sample(x=1:V, size=n1, replace=TRUE)
out <- NULL
for (v in 1:V) {
    train.v <- dat_imputed[id.fold!=v, ]
    test.v <- dat_imputed[id.fold==v, ]

    out <- rbind(out, c(v, NROW(train.v), NROW(test.v)))
}

#
kable(out, linesep="", booktabs=T, align = "c",
      col.names = c("Fold Id", "Training", "Test"),
      caption = "Number of observations allocated to each fold"
      ) %>%
    kable_styling(latex_options = c("HOLD_position"))
```

Fold Id	Training	Test
1	548	67
2	571	44
3	552	63
4	558	57
5	531	84
6	561	54
7	559	56
8	554	61
9	557	58
10	544	71

Table 6: Number of observations allocated to each fold

6 Predictive Modeling

In the steps to follow, we train several classifiers. For each classifier, we train the model by excluding each fold of data and then applying the trained model to that data fold for prediction.

Because of the differing range and variations observed in the predictors, we created this function to conveniently standardize the predictors in training and testing data without repeating codes unnecessarily. This is particularly necessary for the ANN and SVM models.

```
scale_data <- function(train_set, test_set, method=c('standardize', 'normalize')) {
    method <- match.arg(method)
    if (method=='standardize') {

        # standardize predictors
    X.train <- as.data.frame(model.matrix(Category ~ .-1, data = train_set))
    X.test <- as.data.frame(model.matrix(Category ~ .-1, data = test_set))
    X.train.scaled <- scale(X.train)
    X.test.scaled  <- scale(X.test, attributes(X.train.scaled)$`scaled:center`,
                            attributes(X.train.scaled)$`scaled:scale`)
    train.data <- data.frame(Category=train_set$Category, X.train.scaled)

    }

    return(list(train.scaled=train.data, test.scaled=X.test.scaled))
}
```

6.1 Logistic Regression

The best tuning parameter (the LASSO penalty) was determined for each $v \sim \{1, 2, \ldots, V\}$ fold using a 5-fold cross validation on the training data (the combined 9 folds of data) such that error is within 1 standard error of the minimum.

```
library(glmnet)
library(verification)

missclass.error <- double(length = V)
AUC <- double(V)
predicted.outcomes <- observed.target <- NULL
best.lambda_set <- double(V)
for (v in 1:V) {
    train.v <- dat_imputed[id.fold!=v, ]
    test.v <- dat_imputed[id.fold==v, ]
```

```
    # standardize predictors
    scaled.data <- scale_data(train.v, test.v)
    train.scaled <- scaled.data$train.scaled
    X.test.scaled <- scaled.data$test.scaled

    y <- train.v$Category
    X <- train.scaled[,-1]
    X <- model.matrix(~ . -Category, data = train.v)

    # determine best lambda via cross-validation
    cv.lasso <- cv.glmnet(x=X, y=y, family="binomial", alpha = 1, nlambda = 150,
                          standardize = F, thresh = 1e-07, maxit=1000)

   best.lambda <- cv.lasso$lambda.1se #best.lambda
   best.lambda_set[v] <- best.lambda
   fit.best.lambda <-  glmnet(x=X, y=y, family="binomial", alpha = 1,
                  lambda=best.lambda, standardize = F,
                  thresh = 1e-07, maxit=1000)

   # obtain the significant predictors for the final model using a 0 cutoff
   beta.coeff <-as.vector(coef(fit.best.lambda))
   terms <- colnames(X)[abs(beta.coeff[-1]) > 0]
   vars.selected <- stringr::str_remove(terms, "f|m") # remove trailing m or f
                                                      # just in case Sex is selected.
   formula.lasso <- as.formula(paste(c("Category ~ 1", vars.selected),
                                     collapse = "  + "))
   final.mod <- glm(formula.lasso, family = "binomial", data = train.v)

   # make predictions
  pred.prob <- predict(final.mod, newdata=test.v)
  yhat <- ifelse(pred.prob > 0.5, 1, 0)

  # compute AUC
  yobs <- test.v$Category
  AUC[v] <- roc.area(obs = yobs, pred = pred.prob)$A

  # compute  misclassification rate
  missclass.error[v] <- mean(yobs!=yhat)
  # record predicted outcomes along with the observed response
  predicted.outcomes <- c(predicted.outcomes, yhat)
  observed.target <- c(observed.target, yobs)
}

# average AUC
AUC_logistic <- mean(AUC)
# compute the overall misclassification rate
missclass_logistic <- mean(observed.target!=predicted.outcomes)

print(paste("Average AUC:", AUC_logistic))
```

```
## [1] "Average AUC: 0.944087339616688"
```

```
print(paste("Overall missclassification rate:", missclass_logistic))
```

```
## [1] "Overall missclassification rate: 0.0601626016260163"
```

The average AUC values across V folds for the **regularized logistic regression** model is 0.944, while the overall

missclassification rate obtained is 0.0602. By these results, we can tell that the model has a good predictive performance but we shall wait until the end to see how this performance compares to the other models.

Reporting one final logistic model

We report one final logistic model corresponding to the highest AUC for interpretation purposes.

```
# recover the best model during the 10-fold modeling process as a final model
# for reporting (get the fold which gave the highest AUC)
best.v <- which.max(AUC)
train.v <- dat_imputed[id.fold!=best.v, ]
test.v <- dat_imputed[id.fold==best.v, ]
y <- train.v$Category
X <- model.matrix(~ . -Category, data = train.v)
fit.best <-  glmnet(x=X, y=y, family="binomial", alpha = 1,
                    lambda=best.lambda_set[best.v], standardize = F,
                    thresh = 1e-07, maxit=1000)
beta.coeff <- as.vector(coef(fit.best))
terms <- colnames(X)[abs(beta.coeff[-1]) > 0]; terms
```

```
## [1] "ALP"  "ALT"  "AST"  "BIL"  "CREA" "GGT"  "PROT"
```

The best tuning parameter λ was obtained as $\lambda = 0.1288$ based on cross-validation such that error is within 1 standard error of the minimum. Using this optimal tuning parameter for the LASSO penalty, the 7 predictors shown in the above output were the ones selected when a zero cutoff is applied on the estimated coefficients in absolute terms. We then fit the final model corresponding to these selected predictors.

```
formula.lasso <- as.formula(paste(c("Category ~ 1", terms),
                                  collapse = "  + "))
final.logistic <- glm(formula.lasso, family = "binomial", data = train.v)

final.logistic %>% broom::tidy(conf.int=T,conf.level=0.95) %>%
    kable(booktabs=T, linesep="", caption = "Parameter estimates for the final Logistic model")%>%
    kable_styling(latex_options =c("HOLD_position"))
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	-14.9991	3.3872	-4.428	0.0000	-22.1491	-8.8171
ALP	-0.0379	0.0115	-3.286	0.0010	-0.0597	-0.0161
ALT	-0.0251	0.0093	-2.702	0.0069	-0.0437	-0.0081
AST	0.1040	0.0171	6.082	0.0000	0.0722	0.1393
BIL	0.0696	0.0244	2.848	0.0044	0.0199	0.1164
CREA	0.0163	0.0052	3.147	0.0017	0.0070	0.0271
GGT	0.0183	0.0050	3.672	0.0002	0.0086	0.0283
PROT	0.1251	0.0413	3.026	0.0025	0.0489	0.2124

Table 7: Parameter estimates for the final Logistic model

Based on the results in Table 7, we can conclude at 5% significance level that all the predictors are statistically significant in the model since their p-values are smaller than 0.05. We learn from the signs of the coefficients that `ALP` and `ALT` affects the target negatively while the remaining five tend to have a positive effect on the target.

Next, we report the associated odds ratio of the estimated coefficients in 8 with 95% confidence intervals. We can interpret each one of the odds ratios for all the predictors since they are all significant (corresponding CIs do not contain 1). For example, the estimated odds ratio for `AST` is $e^{0.1040} = 1.1096$, which means that for every unit increase in AST, the odds (likelihood) of a donor *being a Hepatitis C patient* increases by a factor of 0.1096, holding all other factors constant. However, every unit increase in ALP decreases the incidence of Hepatitis C by 0.9629 when all other factors are held constant.

```
exp(cbind(OR = coef(final.logistic), confint(final.logistic))) %>%
    kable(booktabs=T, linesep="", caption =
    "Odds ratio based on parameter estimates from the logistic model") %>%
    kable_styling(latex_options =c("HOLD_position"))
```

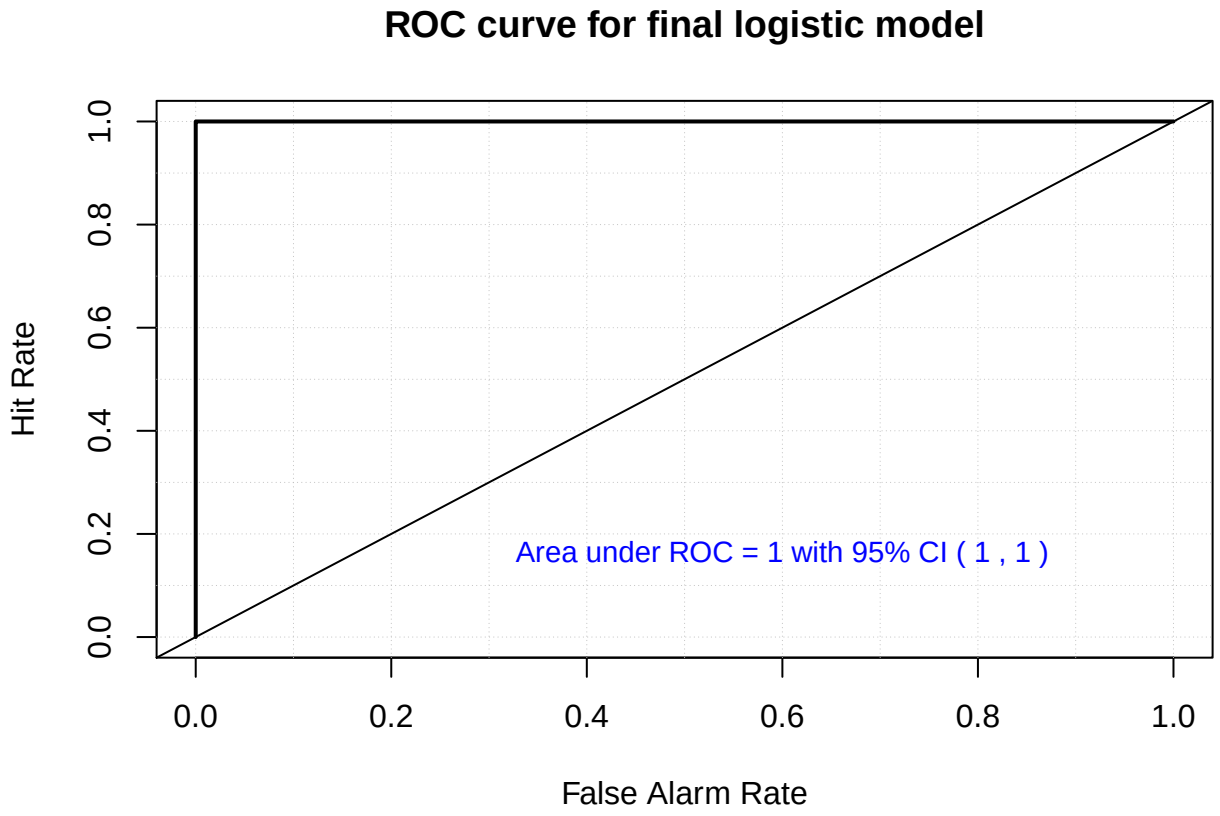
	OR	2.5 %	97.5 %
(Intercept)	0.0000	0.0000	0.0001
ALP	0.9629	0.9421	0.9840
ALT	0.9752	0.9573	0.9920
AST	1.1096	1.0749	1.1494
BIL	1.0721	1.0201	1.1234
CREA	1.0164	1.0070	1.0275
GGT	1.0185	1.0086	1.0287
PROT	1.1332	1.0501	1.2367

Table 8: Odds ratio based on parameter estimates from the logistic model

6.1.1 Area under ROC for the final logistic model

```
library(verification)
library(cvAUC)

phat.logit <- predict(final.logistic, newdata = test.v, type="response") # predicted probabilities

# a custom function for computing and plotting ROC curve
roc_curve <- function (phat, main = "", col="blue", roc_val=F, test.df=test.v) {
   yobs <- test.df$Category
   AUC <- ci.cvAUC(predictions=phat, labels=yobs, folds=1:NROW(test.df),
                confidence=0.95)

   if(roc_val) return(AUC$cvAUC)

   auc.ci <- round(AUC$ci, digits=3) # confidence interval for cross-validated
                                     # Area Under the ROC Curve
   mod <- verify(obs=yobs, pred=phat)
   roc.plot(mod, plot.thres = NULL, main=main)
  text(x=0.6, y=0.16, paste("Area under ROC =", round(AUC$cvAUC, digits=3),
    "with 95% CI (", auc.ci[1], ",", auc.ci[2], ")",
    sep=" "), col=col, cex=.9)
}

roc_curve(phat.logit, main="ROC curve for final logistic model")
```

```
## If baseline is not included, baseline values  will be calculated from the  sample obs.
```



The Area under the ROC curve is impressively **1**, indicating that the best model corresponding to fold 2 made perfect predictions.

6.2 Random Forest (RF): Another baseline model

```
library(randomForest)

missclass.error <- double(length = V)
AUC <- double(V)
predicted.outcomes <- observed.target <- NULL
for (v in 1:V) {
    train.v <- dat_imputed[id.fold!=v, ]
    test.v <- dat_imputed[id.fold==v, ]
    fit.rf <- randomForest(factor(Category) ~., data=train.v, importance=T, ntree=500)
    # get predicted probabilities
    pred.prob.rf <- predict(fit.rf, newdata=test.v, type="prob")[, 2]

    # compute AUC
  yobs <- test.v$Category
  AUC[v] <- roc.area(obs = yobs, pred = pred.prob.rf)$A

  # compute  misclassification rate
  yhat <- ifelse(pred.prob.rf > 0.5, 1, 0)
  missclass.error[v] <- mean(yobs!=yhat)
  # record predicted outcomes along with the observed response
  predicted.outcomes <- c(predicted.outcomes, yhat)
  observed.target <- c(observed.target, yobs)

}
```

```
# average AUC
AUC.rf <- mean(AUC); AUC.rf
```

```
## [1] 0.9806
```

```
# compute the overall misclassification rate
missclass.rf <- mean(observed.target!=predicted.outcomes); missclass.rf
```

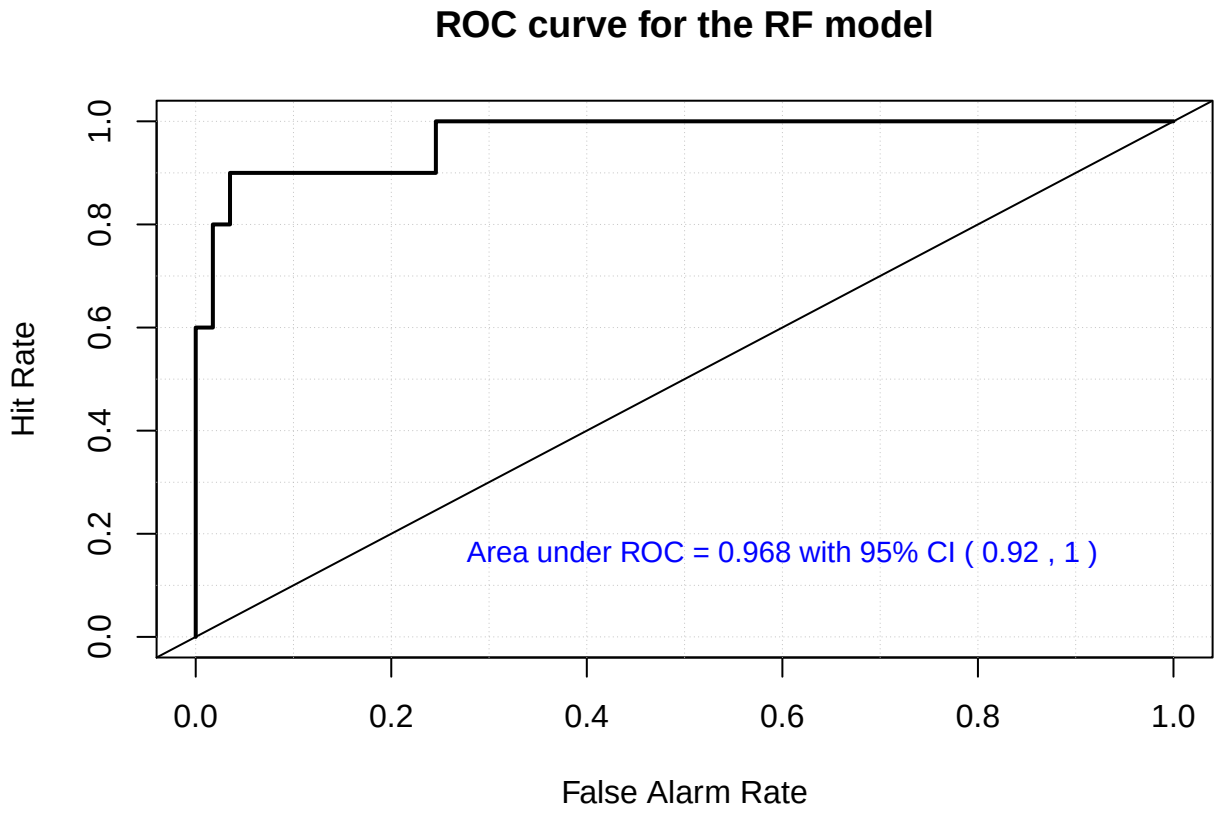
```
## [1] 0.03252
```

The average AUC values across 10 folds for the **random forest** model is 0.981, while the overall missclassification rate obtained is 0.0325. By these results, we can tell that the model has a great predictive performance but it remains to see how this compares to the other models.

6.2.1 Presenting one Final Random Forest Model

```
# recover the best model during the 10-fold modeling process as a final model
# for reporting (get the fold which gave the highest AUC)
best.v <- which.max(AUC.rf)
train.v <- dat_imputed[id.fold!=best.v, ]
test.v <- dat_imputed[id.fold==best.v, ]
fit.rf <- randomForest(factor(Category) ~., data=train.v, importance=T, ntree=500)
    # get predicted probabilities
pred.prob.rf <- predict(fit.rf, newdata=test.v, type="prob")[, 2]

# create ROC curve
roc_curve(pred.prob.rf, main="ROC curve for the RF model")
```

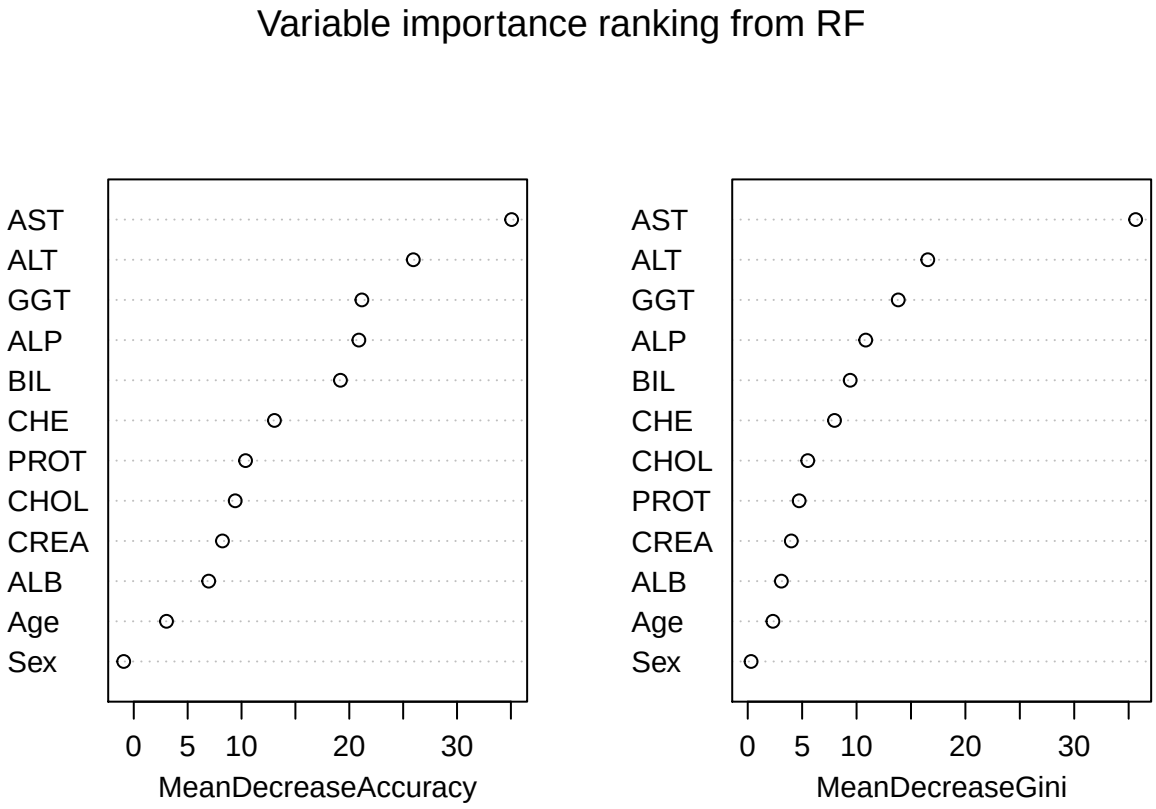
```
## If baseline is not included, baseline values  will be calculated from the  sample obs.
```



The AUC value of **0.968** shows that the random forest model has very high predictive performance for the problem at hand.

6.2.2 Variable Importance Ranking from Random Forest

```
varImpPlot(fit.rf, main = "Variable importance ranking from RF", cex=0.89)
```



According to the mean decrease accuracy, the top four variables are `AST`, `ALT`, `GGT`, and `ALP`. AST turns out to be the most important determinant Hepatitis C incidence. Even based on the mean decrease GINI, AST remains the most influential predictor.

7 Multivariate Adaptive Regression Splines (MARS)

We allowed maximum degree of interactions up to 3 by setting `degree` to 3 for the MARS model.

```
library(earth)

missclass.error <- double(length = V)
AUC <- double(V)
predicted.outcomes <- observed.target <- NULL
for (v in 1:V) {
    train.v <- dat_imputed[id.fold!=v, ]
    test.v <- dat_imputed[id.fold==v, ]
    fit.mars <- earth(Category ~ ., data=train.v, degree=3,
                  glm=list(family=binomial(link = "logit")))

    # get predicted probabilities
    pred.prob.mars <- predict(fit.mars, newdata=test.v, type="response")

    # compute AUC
  yobs <- test.v$Category
  AUC[v] <- roc.area(obs = yobs, pred = pred.prob.mars)$A

  # compute  misclassification rate
```

```
  yhat <- ifelse(pred.prob.mars > 0.5, 1, 0)
  missclass.error[v] <- mean(yobs!=yhat)
  # record predicted outcomes along with the observed response
  predicted.outcomes <- c(predicted.outcomes, yhat)
  observed.target <- c(observed.target, yobs)

}

# average AUC
AUC.mars <- mean(AUC); AUC.mars
```

```
## [1] 0.9173
```

```
# compute the overall misclassification rate
missclass.mars <- mean(observed.target!=predicted.outcomes); missclass.mars
```

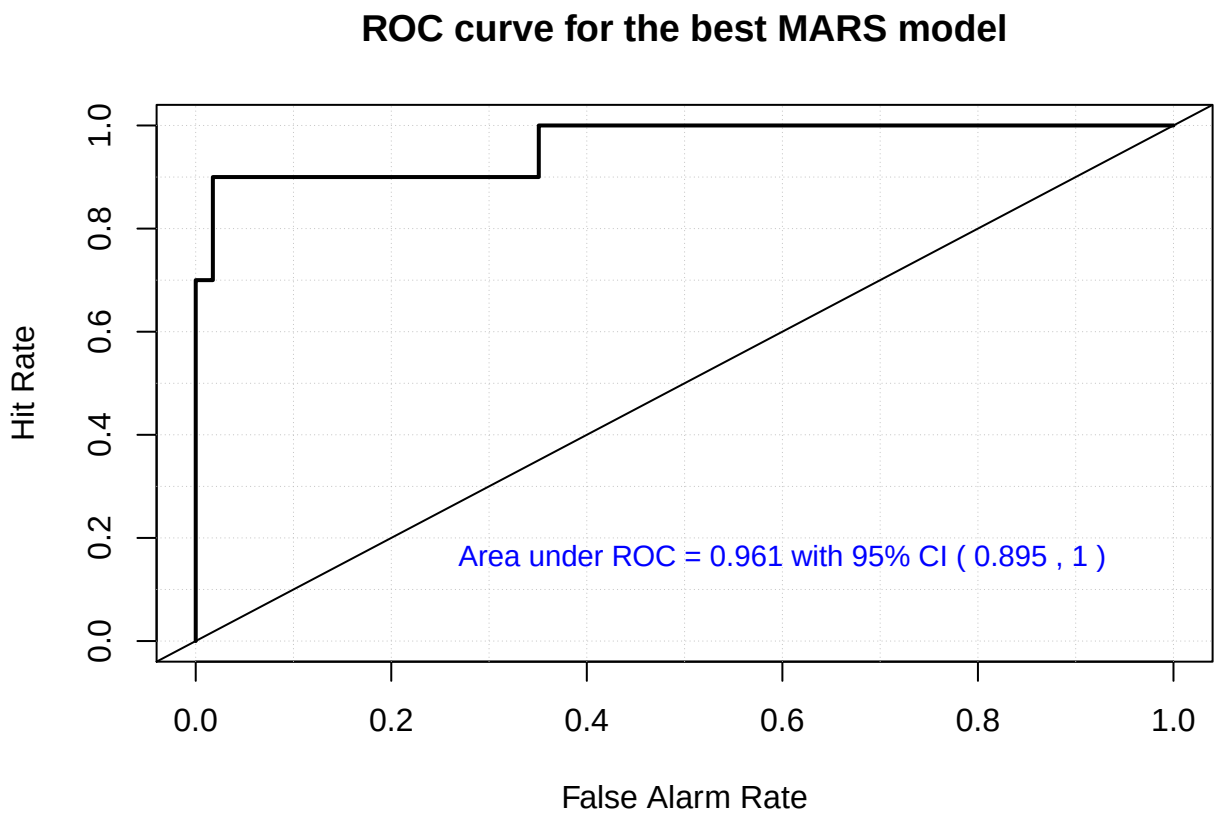
```
## [1] 0.04065
```

The average AUC values across 10 folds for the **MARS** model is 0.917, while the overall missclassification rate obtained is 0.0407. By these results, we can tell that the model has a good predictive performance but not better than the performances of the models already reported.

7.0.1 Presenting one Final MARS Model

```
# recover the best model during the 10-fold modeling process as a final model
# for reporting (get the fold which gave the highest AUC)
best.v <- which.max(AUC.mars)
train.v <- dat_imputed[id.fold!=best.v, ]
test.v <- dat_imputed[id.fold==best.v, ]
fit.mars <- earth(Category ~ ., data=train.v, degree=3,
                  glm=list(family=binomial(link = "logit")))
    # get predicted probabilities
pred.prob.mars <- predict(fit.mars, newdata=test.v, type="response")

# create ROC curve
roc_curve(pred.prob.mars, main="ROC curve for the best MARS model")
```

```
## If baseline is not included, baseline values  will be calculated from the  sample obs.
```



7.1 Variable importance ranking

```
library(vip)
# generating variable importance plot
vip(fit.mars, num_features = 10, aesthetics = list(fill="dodgerblue",
                                                   color="dodgerblue")) +
    ggtitle("Variable importance (GCV) ranking from MARS") +
    scale_fill_brewer(palette = "Set1") +
    theme(plot.title = element_text(hjust = .5))
```

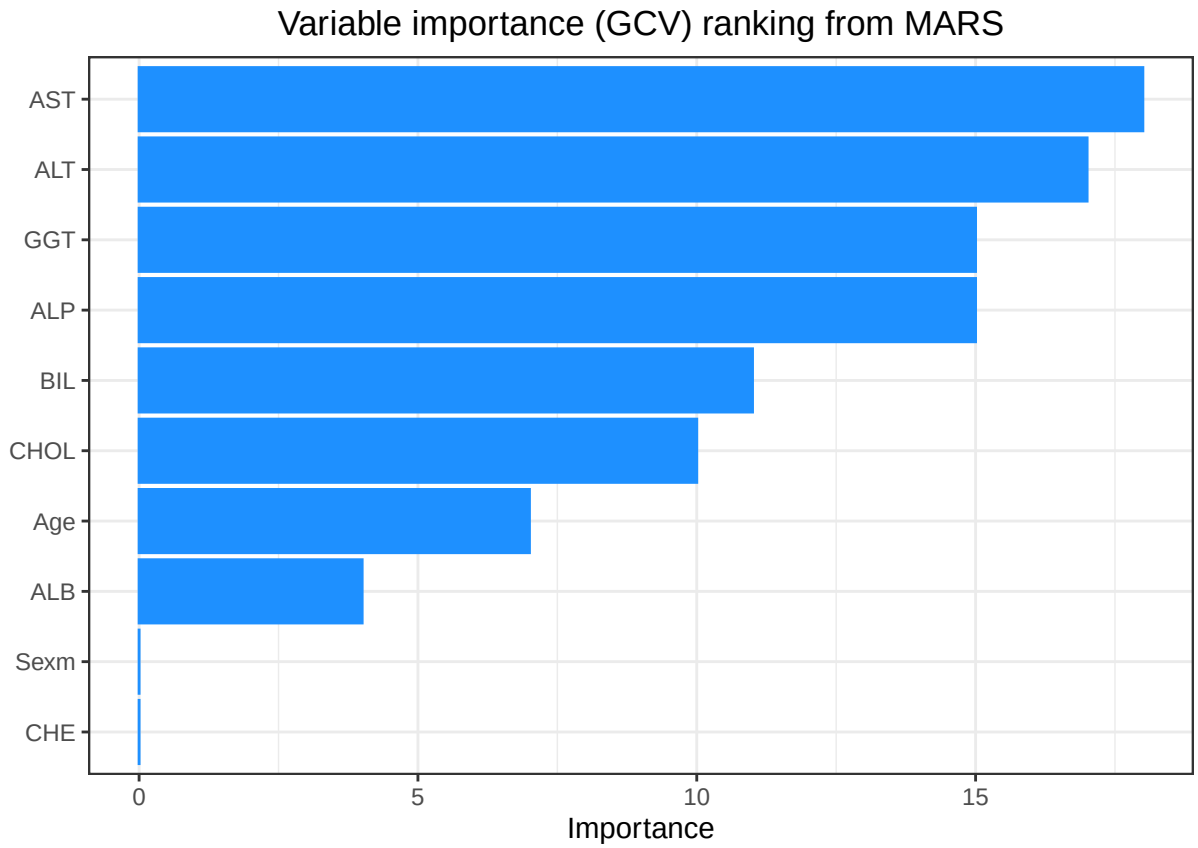


Figure 6: Variable importance based on impact to GCV as predictors are added to the model

We see here that the top four important variables include the top four variables are `AST`, `ALT`, `GGT`, and `ALP`, with `AST` leading the way as in the case of the random forest model.

8 Artificial Neural Network (ANN)

8.0.1 ANN model 1: 1 layer ANN MLP model with 3 hidden units

```
library(neuralnet)
set.seed(125)
missclass.error <- double(length = V)
AUC <- double(V)
predicted.outcomes <- observed.target <- NULL
for (v in 1:V) {
    train.v <- dat_imputed[id.fold!=v, ]
    test.v <- dat_imputed[id.fold==v, ]

    # standardize predictors
    # scaled.data <- scale_data(train.v, test.v)
    # train.scaled <- scaled.data$train.scaled
    # X.test.scaled <- scaled.data$test.scaled
    # # train ANN MLP model
    # fit1.ann <- neuralnet(Category ~ ., data=train.scaled, hidden = 3, rep = 5,
    #                act.fct = "logistic", err.fct = "ce",
    #                linear.output = F, likelihood = T)

    # normalize predictors
```

```
    X.train <- as.data.frame(model.matrix(Category~.,data =train.v))
    X.train$`(Intercept)` <- NULL
    X.test <- as.data.frame(model.matrix(Category~.,data =test.v))
    X.test$`(Intercept)` <- NULL
    X.train.scaled <- scale(X.train)
    X.test.scaled <- scale(X.test,
                        attributes(X.train.scaled)$`scaled:center`,
                        attributes(X.train.scaled)$`scaled:scale`)
    train.data <- data.frame(cbind(Category=train.v$Category, X.train.scaled))
    # train ANN MLP model
    fit1.ann <- neuralnet(Category~.,data=train.data,hidden =3,rep =5,
                        act.fct ="logistic",err.fct ="ce",linear.output =F,likelihood =T)

    # get predicted probabilities
    pred.prob.ann1 <- compute(fit1.ann, covariate = X.test.scaled, rep = 5)$net.result
    # compute AUC
     yobs <- test.v$Category
     AUC[v] <- roc.area(obs = yobs, pred = pred.prob.ann1)$A

      # compute  misclassification rate
     yhat <- ifelse(pred.prob.ann1 > 0.5, 1, 0)
     missclass.error[v] <- mean(yobs!=yhat)
      # record predicted outcomes along with the observed response
     predicted.outcomes <- c(predicted.outcomes, yhat)
     observed.target <- c(observed.target, yobs)

}

# average AUC
AUC.ann1 <- mean(AUC); AUC.ann1
```

```
## [1] 0.9612
```

```
# compute the overall misclassification rate
missclass.ann1 <- mean(observed.target!=predicted.outcomes); missclass.ann1
```

```
## [1] 0.05041
```

8.0.2 Presenting one final ANN Model

The architecture of this model is depicted below.

```
# report the ANN corresponding to the last fold
# plot the model architecture
plot(fit1.ann, rep="best", show.weights=T, dimension=6.5, information=F, radius=.15,
    col.hidden="red", col.hidden.synapse="black", lwd=1, fontsize=9)
```

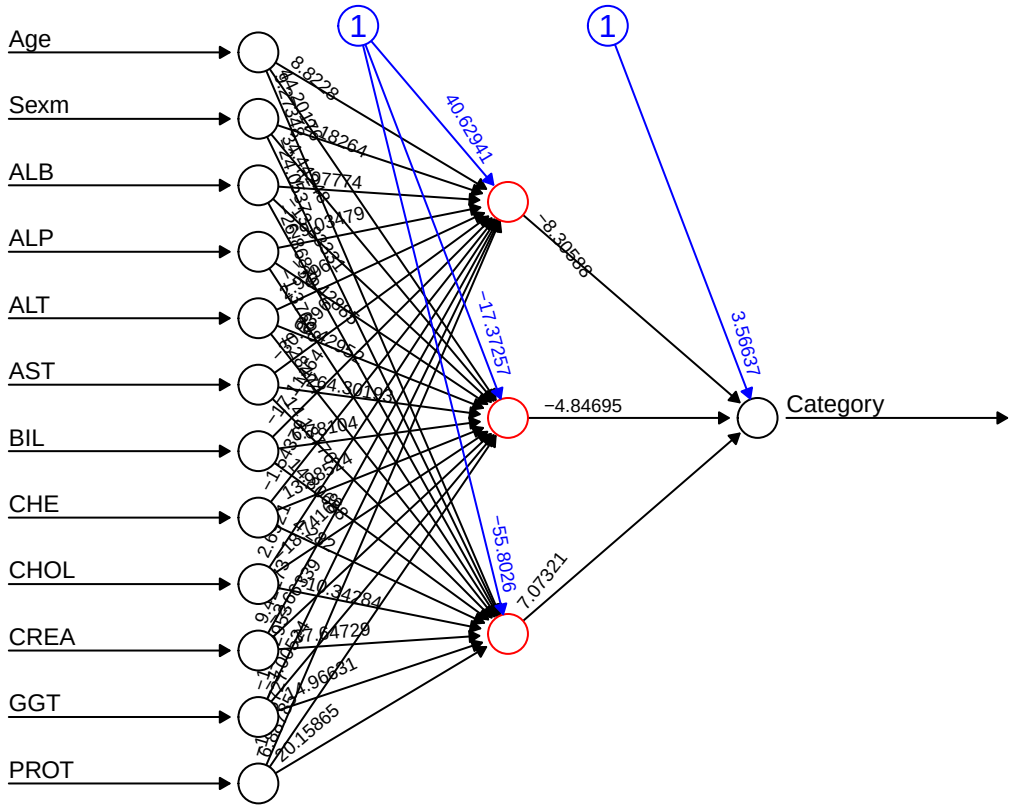
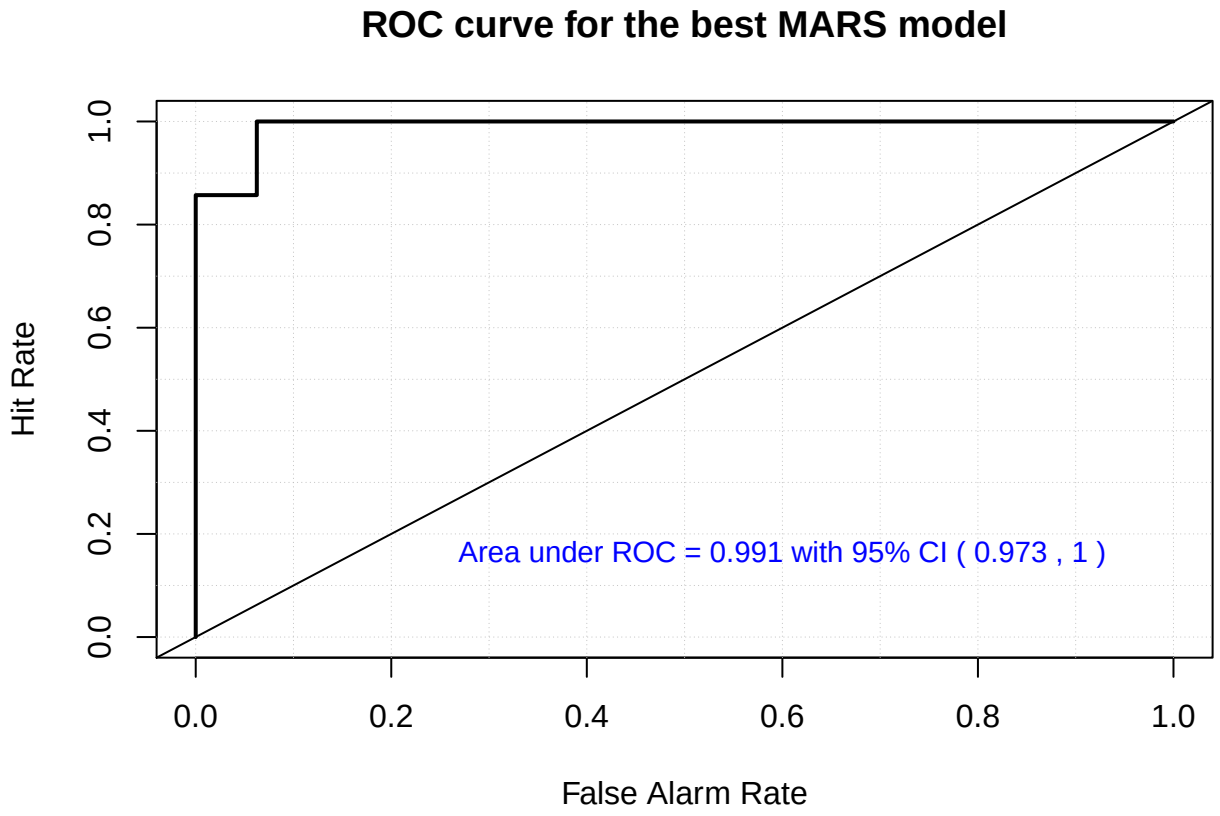


Figure 7: ANN architecture with 1 hiden layer having 3 hidden units

```
    # get predicted probabilities
```

```
# create ROC curve
roc_curve(pred.prob.ann1, main="ROC curve for the best MARS model")
```

```
## If baseline is not included, baseline values  will be calculated from the  sample obs.
```



8.0.3 Model 2: 2 layer ANN MLP model with 2 hidden units in the first layer and 3 hidden units in the second layer.

```
set.seed(125)
missclass.error <- double(length = V)
AUC <- double(V)
predicted.outcomes <- observed.target <- NULL
for (v in 1:V) {
    train.v <- dat_imputed[id.fold!=v, ]
    test.v <- dat_imputed[id.fold==v, ]

    # standardize predictors
    X.train <- as.data.frame(model.matrix(Category ~ .-1, data = train.v))
    X.test <- as.data.frame(model.matrix(Category ~ .-1, data = test.v))
    X.train.scaled <- scale(X.train)
    X.test.scaled  <- scale(X.test, attributes(X.train.scaled)$`scaled:center`,
                            attributes(X.train.scaled)$`scaled:scale`)
    train.data <- data.frame(cbind(Category=train.v$Category, X.train.scaled))


    # train ANN MLP model
    fit2.ann <- neuralnet(Category ~ ., data=train.data, hidden = c(4, 3), rep = 5,
                  act.fct = "logistic", err.fct = "ce",
                  linear.output = F, likelihood = T)

    # get predicted probabilities
    pred.prob.ann2 <- compute(fit2.ann, covariate = X.test.scaled, rep = 5)$net.result

    # compute AUC
  yobs <- test.v$Category
```

```
  AUC[v] <- roc.area(obs = yobs, pred = pred.prob.ann2)$A

  # compute  misclassification rate
  yhat <- ifelse(pred.prob.ann2 > 0.5, 1, 0)
  missclass.error[v] <- mean(yobs!=yhat)
  # record predicted outcomes along with the observed response
  predicted.outcomes <- c(predicted.outcomes, yhat)
  observed.target <- c(observed.target, yobs)

}

# average AUC
AUC.ann2 <- mean(AUC); AUC.ann2
```

```
## [1] 0.9581
```

```
# compute the overall misclassification rate
missclass.ann2 <- mean(observed.target!=predicted.outcomes); missclass.mars
```

```
## [1] 0.04065
```

8.0.4 Presenting Final ANN Model

```
# report the ANN corresponding to the last fold

# plot the model architecture
plot(fit2.ann, rep="best", show.weights=T, dimension=6.5, information=F, radius=.15,
    col.hidden="red", col.hidden.synapse="black", lwd=1, fontsize=9)
```

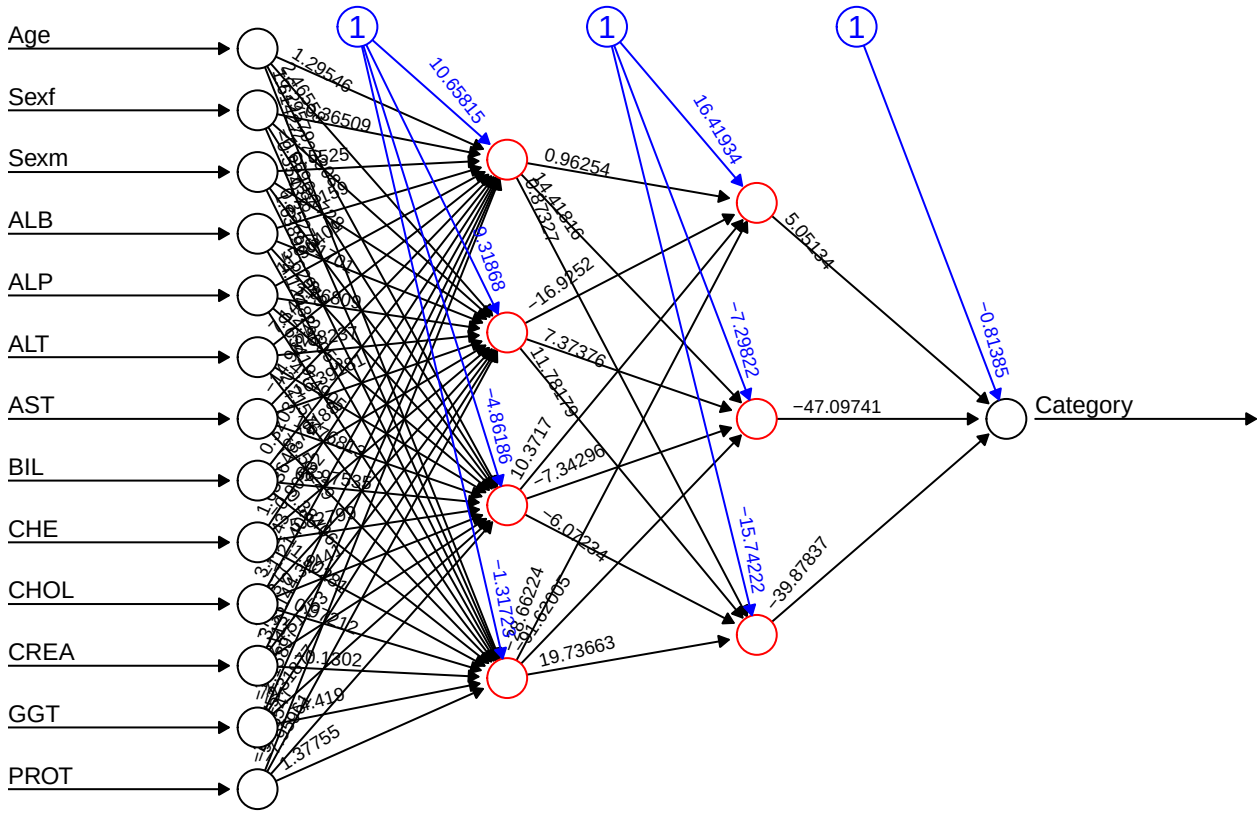
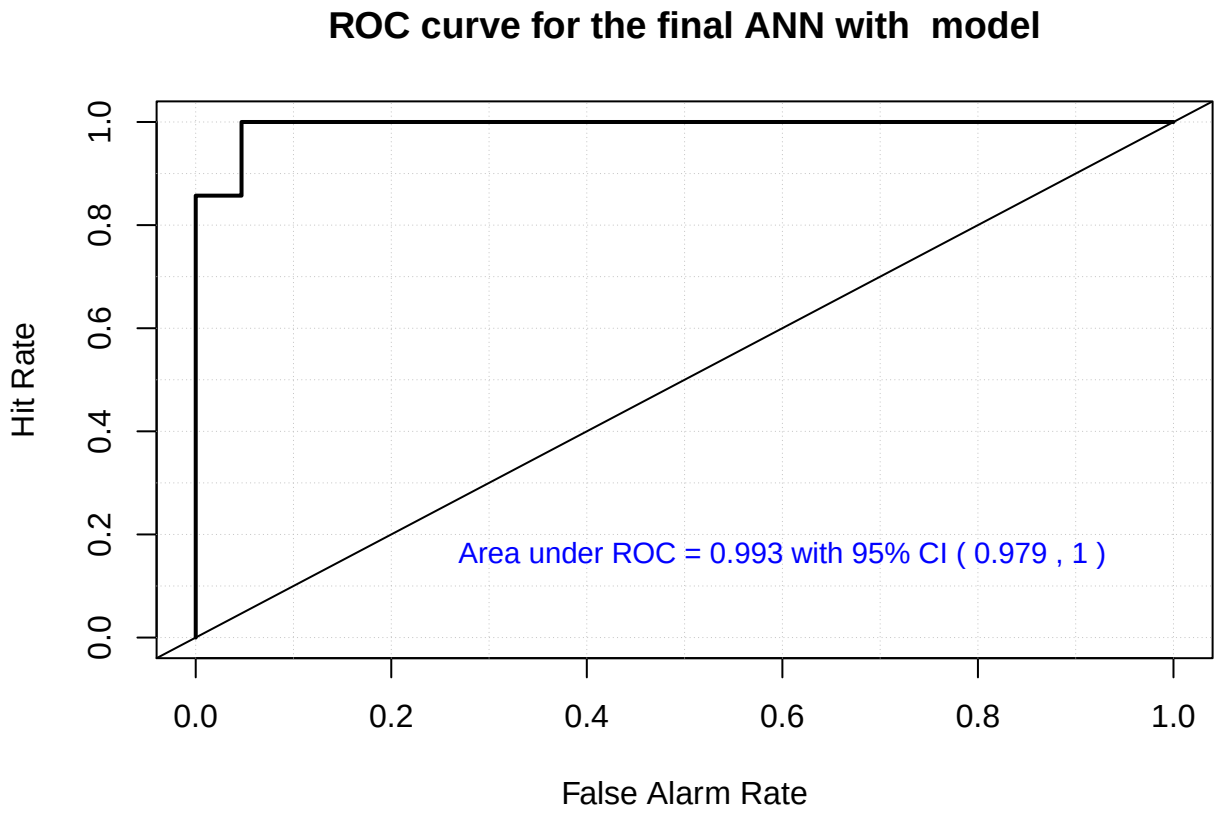


Figure 8: ANN architecture with 2 hiden layers having 4 and 3 hidden units, respectively.

```
    # get predicted probabilities
```

```
# create ROC curve
roc_curve(pred.prob.ann2, main="ROC curve for the final ANN with  model")
```

```
## If baseline is not included, baseline values  will be calculated from the  sample obs.
```



8.1 Support Vector Machines (SVM)

8.2 SVM Linear

```
library(caret)

missclass.error <- double(length = V)
AUC <- double(V)
predicted.outcomes <- observed.target <- NULL

# assign meaningful levels to the target  to prevent an error
# we run into when classProbs was set to TRUE
dat_imputed2 <- dat_imputed %>%
    mutate(Category = factor(if_else(Category==0, "Healthy", "Unhealthy")))

for (v in 1:V) {
    train.v <- dat_imputed2[id.fold!=v, ]
    test.v <- dat_imputed[id.fold==v, ]


    # standardize predictors
    scaled.data <- scale_data(train.v, test.v)
    train.scaled <- scaled.data$train.scaled
```

```
    X.test.scaled <- scaled.data$test.scaled

    # selecting the optimal C
    grid <- expand.grid(C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25,
                              1.5, 1.75, 2,5))
    fit.svmLinear <- train(Category ~., data = train.scaled,
                           method = "svmLinear",
                       trControl=trainControl(method = "repeatedcv", number=5,
                                              classProbs = T, repeats=3,
                                              summaryFunction = twoClassSummary),
                       # preProcess = c("center", "scale"),
                       tuneGrid = grid, tuneLength = 10)

    # get predicted probabilities
    pred.prob.svmLinear <- predict(fit.svmLinear, newdata=X.test.scaled,
                                   type = "prob")[, "Unhealthy"]

    # compute AUC
  yobs <- test.v$Category
  AUC[v] <- roc.area(obs = yobs, pred = pred.prob.svmLinear)$A

  # compute  misclassification rate
  yhat <- ifelse(pred.prob.svmLinear > 0.5, 1, 0)
  missclass.error[v] <- mean(yobs!=yhat)
  # record predicted outcomes along with the observed response
  predicted.outcomes <- c(predicted.outcomes, yhat)
  observed.target <- c(observed.target, yobs)

}

# average AUC
AUC.svmLinear <- mean(AUC); AUC.svmLinear
```

```
## [1] 0.9593
```

```
# compute the overall misclassification rate
missclass.svmLinear <- mean(observed.target!=predicted.outcomes); missclass.svmLinear
```

```
## [1] 0.05203
```

8.2.1 Presenting one final SVM Linear Model

The result below provides information about the chosen SVM linear model revealing the best values of the tuning parameters used.

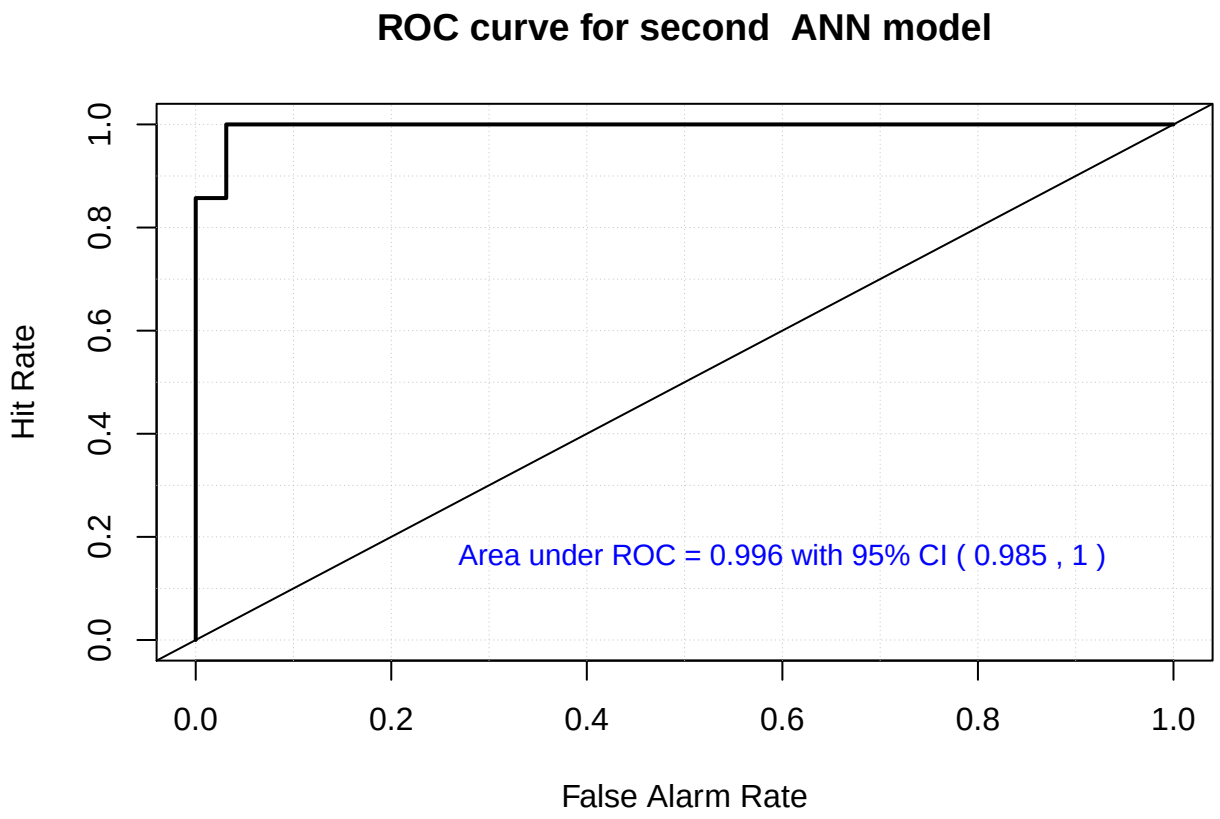
```
# model information
fit.svmLinear$finalModel
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 0.5
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 71
##
## Objective Function Value : -30
```

```
## Training error : 0.036765
## Probability model included.
```

```
# create ROC curve
roc_curve(pred.prob.svmLinear, main="ROC curve for second  ANN model")
```

```
## If baseline is not included, baseline values  will be calculated from the  sample obs.
```



8.3 SVM Radial (Non-linear kernel)

```
missclass.error <- double(length = V)
AUC <- double(V)
predicted.outcomes <- observed.target <- NULL

for (v in 1:V) {
    train.v <- dat_imputed2[id.fold!=v, ]
    test.v <- dat_imputed[id.fold==v, ]


    # standardize predictors
    scaled.data <- scale_data(train.v, test.v)
    train.scaled <- scaled.data$train.scaled
    X.test.scaled <- scaled.data$test.scaled

    fit.svmRadial <- train(Category ~., data = train.scaled,
                           method = "svmRadial",
                       trControl=trainControl(method = "repeatedcv", number=5,
                                              classProbs = T, repeats=3,
                                              summaryFunction = twoClassSummary),
                       tuneLength = 10)
```

```
    # get predicted probabilities
    pred.prob.svmRadial <- predict(fit.svmRadial, newdata=X.test.scaled,
                                   type = "prob")[, "Unhealthy"]

    # compute AUC
  yobs <- test.v$Category
  AUC[v] <- roc.area(obs = yobs, pred = pred.prob.svmRadial)$A

  # compute  misclassification rate
  yhat <- ifelse(pred.prob.svmRadial > 0.5, 1, 0)
  missclass.error[v] <- mean(yobs!=yhat)
  # record predicted outcomes along with the observed response
  predicted.outcomes <- c(predicted.outcomes, yhat)
  observed.target <- c(observed.target, yobs)

}

# average AUC
AUC.svmRadial <- mean(AUC); AUC.svmRadial
```

```
## [1] 0.9796
```

```
# compute the overall misclassification rate
missclass.svmRadial <- mean(observed.target!=predicted.outcomes); missclass.svmRadial
```

```
## [1] 0.04228
```

8.3.1 Presenting a Final SVM Linear Model

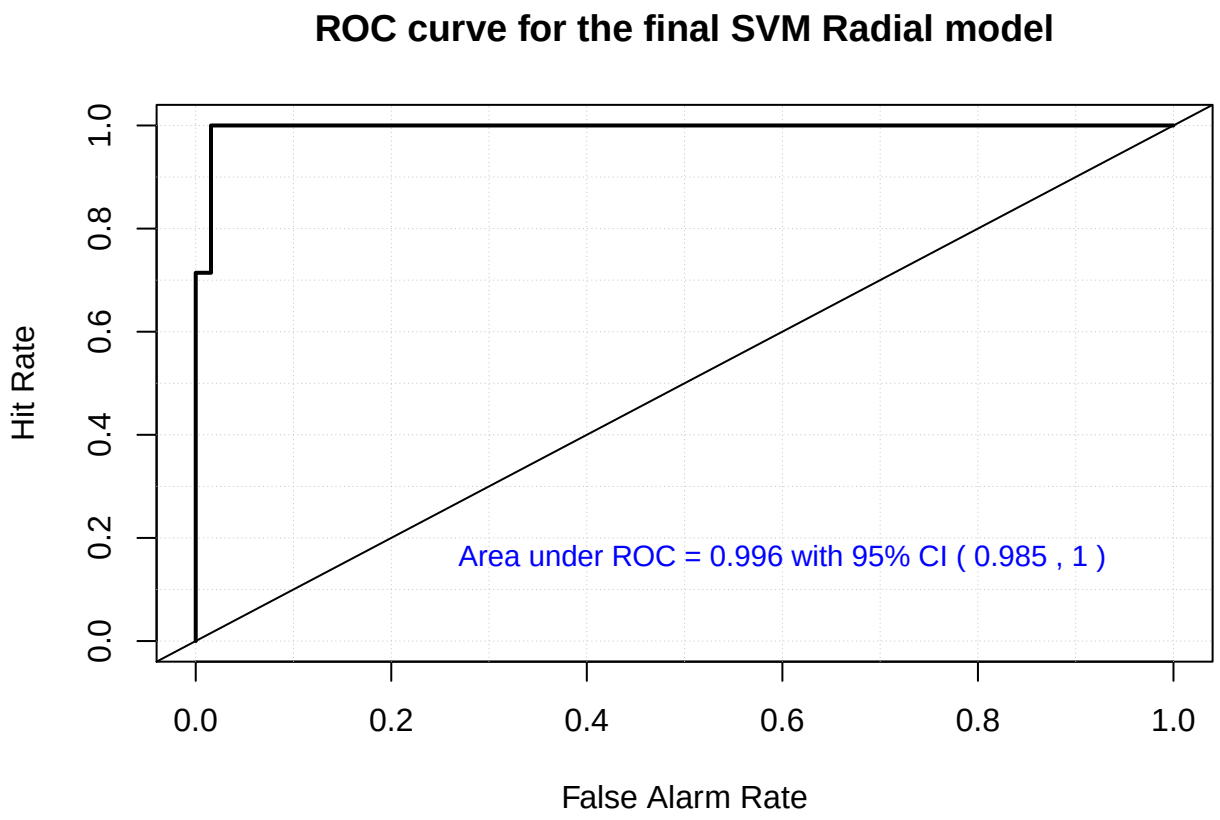
```
# model information
fit.svmRadial$finalModel
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.102334494346306
##
## Number of Support Vectors : 132
##
## Objective Function Value : -57.24
## Training error : 0.016544
## Probability model included.
```

The above output shows information about the chosen SVM model with radial basis kernel function revealing the best values of the tuning parameters used.

```
# create ROC curve
roc_curve(pred.prob.svmRadial, main="ROC curve for the final SVM Radial model")
```

```
## If baseline is not included, baseline values  will be calculated from the  sample obs.
```



8.4 Summary of Results and Model Comparison

```
auc_values <- c(AUC_logistic, AUC.rf, AUC.mars, AUC.ann1, AUC.ann2,
                AUC.svmLinear, AUC.svmRadial)

missclass_values <- c(missclass_logistic, missclass.rf, missclass.mars, missclass.ann1,
                      missclass.ann2, missclass.svmLinear, missclass.svmLinear)

data.frame("Model"= c("Logistic (LASSO)","Random Forest","MARS","ANN1", "ANN2",
                      "SVM Linear", "SVM Radial"),
           "AUC"= auc_values, "Missclassification Rate"=missclass_values) %>%
 kable(booktabs=T, align = "lcc",
       caption = "Model performance metrics averaged over 10 folds")%>%
    kable_styling(latex_options =c("HOLD_position"))
```

Model	AUC	Missclassification.Rate
Logistic (LASSO)	0.9441	0.0602
Random Forest	0.9806	0.0325
MARS	0.9173	0.0407
ANN1	0.9612	0.0504
ANN2	0.9581	0.0553
SVM Linear	0.9593	0.0520
SVM Radial	0.9796	0.0520

Table 9: Model performance metrics averaged over 10 folds

In general, with over 90% prediction accuracy, the results obtained shows that all the classifiers considered did considerably well. The random forest (RF) model yielded the highest predictive power since it provides the largest

AUC value of 0.9806 and the lowest missclassification rate for determining the likelihood of Hepatitis C incidence, followed closely by the SVM with radial basis function which however had a higher classification error than the worst performing model in terms of AUC, MARS. However, among the main models of interest (MARS, ANN, SVM), SVM radial is the best model when it comes to AUC whiles MARS is the best in terms of missclassification rate. Also, between the two ANN models considered, the one layer ANN with 3 hidden units performed better.

RF has the advantages of being a non-linear classifier requiring no distributional assumptions as in the case of models like the regularized logistic regression. Contrasted with SVM, SVM performs well in scenarios where there is a clear separation between the classes to be predicted, and it is more effective in high dimensional spaces where the dimension exceeds the sample size. Although the regularized logistic regression model did not perform so well here compared to most of the other methods in terms of predictive performance, its estimates lend themselves to a good interpretation. For instance, using the odds ratios obtained from the logistic coefficients, we can explain how exactly a particular laboratory value or demographic characteristics such as age influences Hepatitis C incidence. As an example, the estimated odds ratio for `AST` is $e^{0.1040} = 1.1096$, which means that for every unit increase in AST, the odds (likelihood) of a donor *being a Hepatitis C patient* increases by a factor of 0.1096, holding all other factors constant. For this reason, the traditional logistic regression model may be preferred, especially in clinical studies.

ANN on the other hand, can provide a very competitive tool in predictive modeling when the its specification is carefully chosen. This specification, however, takes experience. Another downside worth noting is that, unlike traditional methods such as logistic, the results of an ANN model are difficult to understand and to explain to users.

Random forest and MARS both ranked the laboratory values `AST`, `ALT`, `GGT`, and `ALP` as the top four most significant predictors of Hepatitis C incidence.