Appendix

1.) **Formulas for Model of Endstage Liver Disease (MELD), Kings-Score (KS) and modified Glascow Prognostic Score (mGPS)**
2.) **Analysis of the imputed Data**
3.) **Code of the Workflow for the Calculation of the Proposed Model to Predict Disease Free Survival after Liver Resection for early HCC**

1.) The formulas for the Model of Endstage Liver Disease (MELD), Kings-Score (KS) and modified Glascow Prognostic Score (mGPS)

MELD: 10 x [0,957 x ln(Kreatinin) + 0,378 x ln(Bilirubin) + 1,12 x ln(INR) + 0,643]

KS: Age (years) x AST (U/L) x [INR / Platelet count (109 /L)]

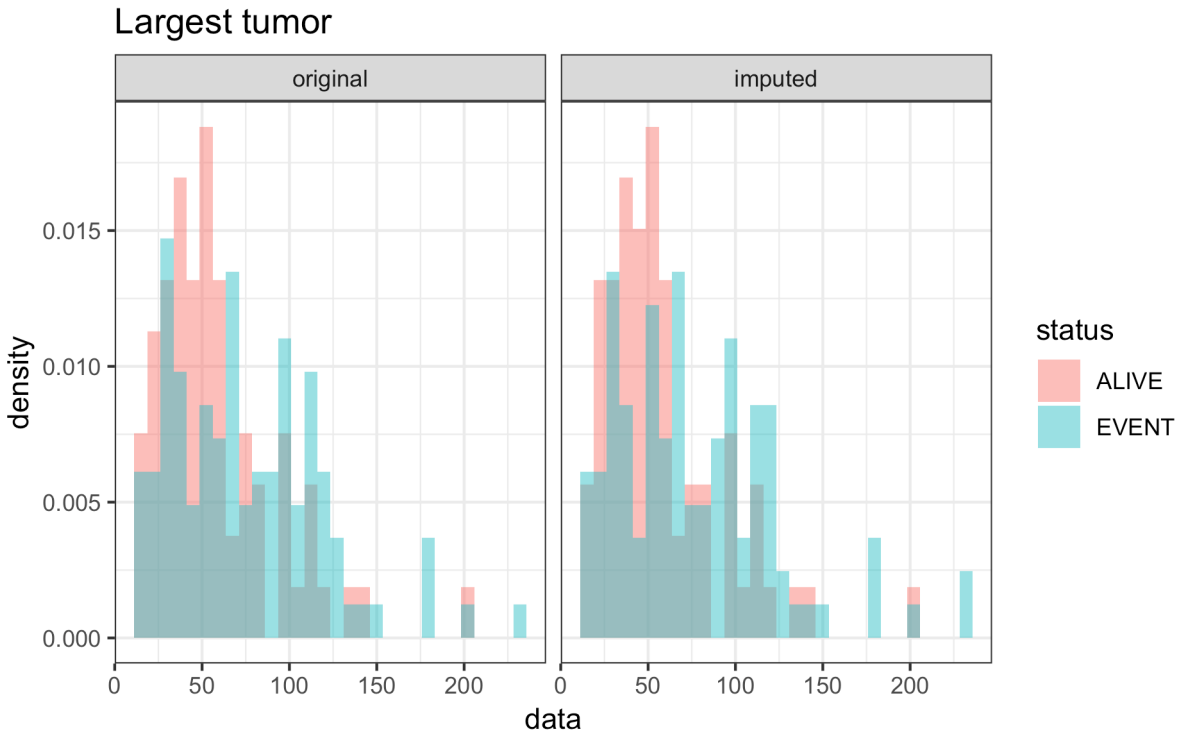
mGPS: CRP ≤ 10 mg/L and albumin ≥ 35 g/L = 0; CRP > 10 mg/L and albumin ≥ 35 g/L = 1; CRP > 10 mg/L and albumin < 35 g/L = 2

2.) Analysis of the imputed Data

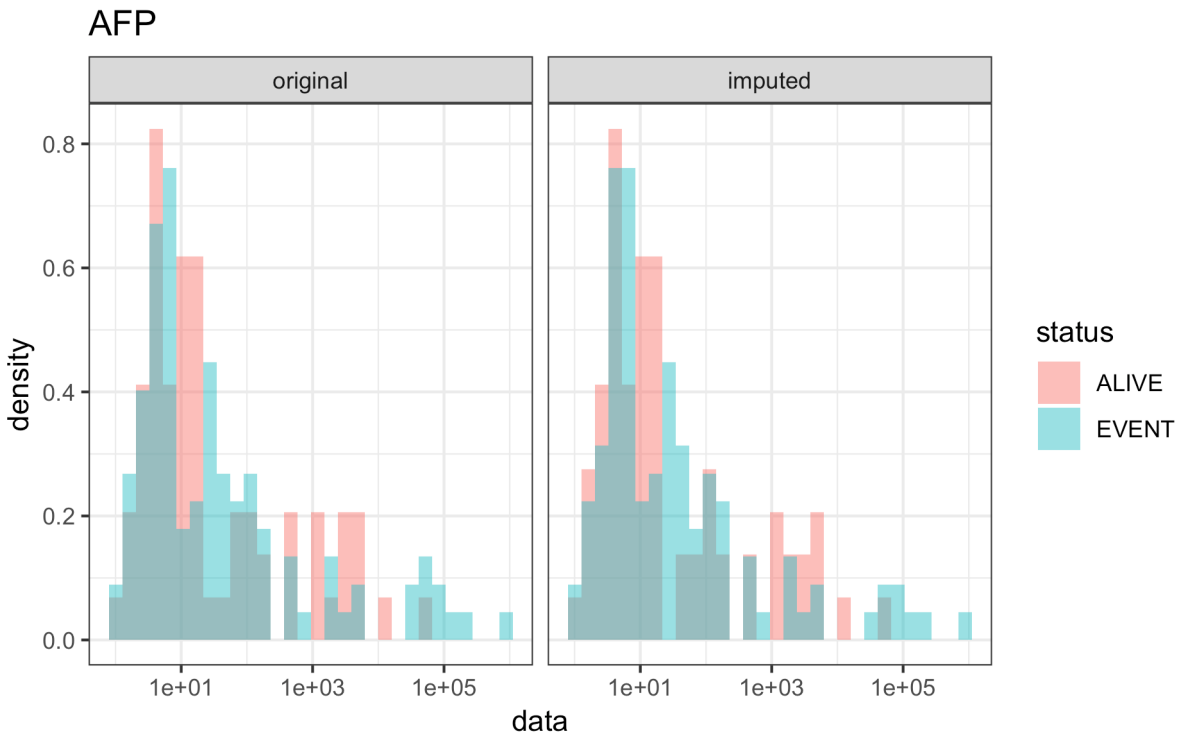
Altogether only two variables showed more than 10% missing values. Largest tumor had 14.9% (n=27) missing values and AFP values were missing in 13.8% (n=25) of cases.

As mentioned in the methods section we used the mice package to impute the values. In figure 1 and 2 the change in distribution for largest tumor and AFP can be observed.

Lastly largest tumor remained as relevant factor after RFE. Therefore, we recalculated the RF model omitting all observations having missing values of largest tumor. The model, which was based on 154 observation reached an AUC of 0.740, which was nearly identical to 0.749 in the original model (without anti-classification) based on all observation.



Suppl Figure 1: Changes in distribution after imputation of missing values in largest tumor



Suppl Figure 2: Changes in distribution after imputation of missing values in AFP

3.) Code of the Workflow for the Calculation of the Proposed Model to Predict Disease Free Survival after Liver Resection for early HCC

1. Preprocessing including imputation of missing values

```
mice_plot <- aggr(data, col=c('navyblue','yellow'),
                  numbers=TRUE, sortVars=TRUE,
                  labels=names(data3), cex.axis=.7, cex.numbers=0.8,
                  gap=3, ylab=c("Missing data","Pattern"))

#If there is one observation with >50% of data na -> discard
#Data imputation
imputed_Data <- mice(data, m=5, maxit = 5, method = 'rf', seed = 500, remove.collinear = F)
summary(imputed_Data)
imputed_Data$loggedEvents
head(imputed_Data$loggedEvents)

#Remove logged events by e.g. remove variables
#Repeat data imputation
imputed_Data <- mice(data, m=5, maxit = 5, method = 'rf', seed = 500, remove.collinear = F)
summary(imputed_Data)
imputed_Data$loggedEvents
head(imputed_Data$loggedEvents)

mice_plot <- aggr(complete(imputed_Data,2), col=c('navyblue','yellow'),
                  numbers=TRUE, sortVars=TRUE,
                  labels=names(data_even_better), cex.axis=.7, cex.numbers=0.8,
                  gap=3, ylab=c("Missing data","Pattern"))

data <- complete(imputed_Data,2)
```

2. Multivariate Analysis of the entire data set

```
#making formulas
formulas.cox <- sapply(c(variables),
                       function(x)as.formula(paste('Surv(follow_up,Class=="EVENT")~',x)))

#making a list of models
models.cox <- lapply(formulas.cox, function(x){coxph(x,data=data)})

#Multivariates Modelling Collet
#Step 1 p<0.2
cox.multi <- coxph(survobj ~ variables, data=data)
summary(cox.multi.recurr)

#Step 2 all p>0.2
cox.multi.step2 <- coxph(survobj ~ variables, data=data)
summary(cox.multi.recurr.step2)
```

```
#Step3 Include all that are <0.1 in Step 1 and 2
cox.multi.final <- coxph(survobj ~ variables, data=data)
summary(cox.multi.recurr.final)
```

3. **Random data partitioning**

```
training <- data$Class %>%
  createDataPartition(p = 0.7, list = FALSE)
test <- data[-training, ]
train  <- data[training, ]
```

4. **Recursive feature elimination (training data)**

```
control <- rfeControl(functions=rfFuncs, method="cv",repeats = 10,verbose = FALSE)
rfe.train <- rfe(train [,1:nVariables], train.data[,OutcomeVariable], sizes=c(1:nVariables), rfeControl=control)
```

- **Clinical curation of selected variables (anti-classification)**

```
#Remove variables deemed protected and reformulate variables to include in train function
rfe_predictors <- predictors(rfe.train)
remove <- c ("protectedVariable1"," protectedVariable2"," protectedVariable3", … protectedVariablen")

rfe_predictors %in% remove

 rfe_predictors_curated <- rfe_predictors [! rfe_predictors %in% remove]

rfe_variables_curated <-
{
  variables <- rfe_predictors_curated
reformulate(termlabels = variables, response = "Class")
}
```

5. **Resampling to balance data (training data) and 6. Random Forest Modelling (training data)**

```
#Resampling can be done by including "sampling" in the control function for caret
fitControl$sampling <- "down", "up" or "SMOTE"
model <- train(
Class ~., data = train, method = "rf",
trControl = fitControl,
importance = TRUE, weights = model_weights
)
```

7. **Prediction of test data and performance measurement**

```
list <- list(original = model,
```

```
                downsampling = model_down,
                 upsampling = model_up,
                 SMOTE = model_smote)

test_ROC <- function(model, data) {
  roc(data$Class,
      predict(model, data, type = "prob")[, "EVENT"])
}

list_roc <- list %>%
  map(test_ROC, data = test)

list_roc %>%
  map(auc)
```