Final Project

Lihui Chen   Xuechen Yu   Richard Li (Xuanyou)

1. Background and Motivation

Metastatic colorectal cancer (mCRC), which is a significant challenge in oncology, remains something that has a substantial impact on patient morbidity and mortality. Even though there have been a lot of advancements in therapeutic challenges, the prognosis for patients with mCRC is often poor. The selection of first-line therapy is thus a critical determinant of clinical outcomes.

The management of mCRC has evolved with the introduction of targeted therapies, which aim to improve survival rates and quality of life. Among these, the epidermal growth factor receptor (EGFR) inhibitor panitumumab has shown promise when used in combination with the conventional chemotherapy regimen FOLFOX (a combination of folinic acid, fluorouracil, and oxaliplatin).

The trial, NCT00364013, is a randomized, multicenter, phase 3 study. It was designed to evaluate the efficacy of panitumumab in combination with FOLFOX versus FOLFOX alone as first-line therapy for patients with previously untreated mCRC. This study holds a repository of rich clinical data, including patient demographics, treatment details, response criteria, survival metrics, and adverse events. Such comprehensive data presents an invaluable opportunity for the application of advanced statistical and machine learning models to predict patient outcomes.

The motivation behind utilizing these computational models lies in their ability to investigate complex patterns within the data. It potentially leads to the identification of prognostic variables and the development of predictive models for patient survival. By applying these methodologies, the research can shed light on the multifaceted nature of mCRC progression and response to treatment. Moreover, it strives to enhance the decision-making process in clinical settings, enabling personalized medicine approaches and optimizing therapeutic efficacy.

2. Research question

1. How can robust statistical and machine learning models be developed and validated using the trial dataset to accurately predict mortality in patients with metastatic colorectal cancer (mCRC)?
2. What are the key clinical and treatment-related factors that significantly impact the survival of patients with mCRC, and how can their influence be quantified and incorporated into predictive models for patient survival?

3. Data cleaning and exploration

```r
library(bis620.2022)
library(readr)
library(dplyr)
#>
```

```r
#> Attaching package: 'dplyr'
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union
library(tidyr)

data <- adsl %>%
  full_join(biomark, by = 'SUBJID')

data1 = data

my_func <- function(df) {

  # Function to convert a categorical column to numeric
  convert_to_numeric <- function(column) {
    # Treat NA or blank values as 'unknown'
    column[is.na(column) | column == ""] <- "unknown"

    # Convert the categorical column to a factor and then to numeric
    as.numeric(factor(column)) - 1  # Subtract 1 to start encoding at 0
  }

  # Create dummy variables for 'sex' and 'race', and bind them to the dataset
  # if("sex" %in% names(data)) {
  #   sex_dummies <- model.matrix(~ sex - 1, data)
  #   colnames(sex_dummies) <- paste("sex", colnames(sex_dummies), sep = "_")
  #   data <- bind_cols(data, as.data.frame(sex_dummies))
  #   data <- data %>% select(-sex)
  # }

  # if("race" %in% names(data)) {
  #   race_dummies <- model.matrix(~ race - 1, data)
  #   colnames(race_dummies) <- paste("race", colnames(race_dummies), sep = "_")
  #   data <- bind_cols(data, as.data.frame(race_dummies))
  #   data <- data %>% select(-race)
  # }

  # Apply the conversion to all other categorical columns except the specified ones
  categorical_columns <- sapply(df, is.character)

  categorical_columns[1] <- FALSE # Assuming the first column is the Subject ID
  #categorical_columns[20:37] <- FALSE # Exclude columns 20 to 37
  categorical_columns[20] <- FALSE
  categorical_columns[22] <- FALSE
  categorical_columns[24] <- FALSE
  categorical_columns[26] <- FALSE
  categorical_columns[28] <- FALSE
  categorical_columns[30] <- FALSE
  categorical_columns[32] <- FALSE
  categorical_columns[34] <- FALSE
```

```r
  categorical_columns[36] <- FALSE

  df[categorical_columns] <- lapply(df[categorical_columns], convert_to_numeric)

  # variables of interest
  var_of_interest <- c("ATRT", "PRSURG", "LIVERMET", "AGE", "SEX", "B_WEIGHT", "B_HEIGHT", "B_ECOG", "B
  df <- df[var_of_interest]

  # Explore missing values
  missing_values <- numeric(length(var_of_interest))

  for (i in seq_along(var_of_interest)) {
    missing_values[i] <- sum(is.na(df[[var_of_interest[i]]]))
  }

  missing_values_df <- data.frame(Variable = var_of_interest, MissingValuesCount = missing_values)
  print(missing_values_df)

  # Remove rows with any missing values
  final_df <- na.omit(df)

  print(paste("Original number of rows:", nrow(df)))
  print(paste("Number of rows after removing missing data:", nrow(final_df)))

  return(final_df)
}

final_df_clean = my_func(data)
#>    Variable MissingValuesCount
#> 1      ATRT                  0
#> 2    PRSURG                  0
#> 3  LIVERMET                  0
#> 4       AGE                  0
#> 5       SEX                  0
#> 6  B_WEIGHT                  0
#> 7  B_HEIGHT                  1
#> 8    B_ECOG                  0
#> 9  B_METANM                  3
#> 10 DIAGTYPE                  0
#> 11      DTH                  0
#> 12    DTHDY                  0
#> [1] "Original number of rows: 935"
#> [1] "Number of rows after removing missing data: 931"
```

4. Analysis

```r
# Check Package Installation
library(randomForest)
#> randomForest 4.7-1.1
#> Type rfNews() to see new features/changes/bug fixes.
#>
```

```r
#> Attaching package: 'randomForest'
#> The following object is masked from 'package:dplyr':
#>
#>     combine
library(rpart.plot)
#> Loading required package: rpart
library(caTools)
library(caret)
#> Loading required package: ggplot2
#>
#> Attaching package: 'ggplot2'
#> The following object is masked from 'package:randomForest':
#>
#>     margin
#> Loading required package: lattice
library(pROC)
#> Type 'citation("pROC")' for a citation.
#>
#> Attaching package: 'pROC'
#> The following objects are masked from 'package:stats':
#>
#>     cov, smooth, var
```

4.1 Survival Analysis – Coxph Model

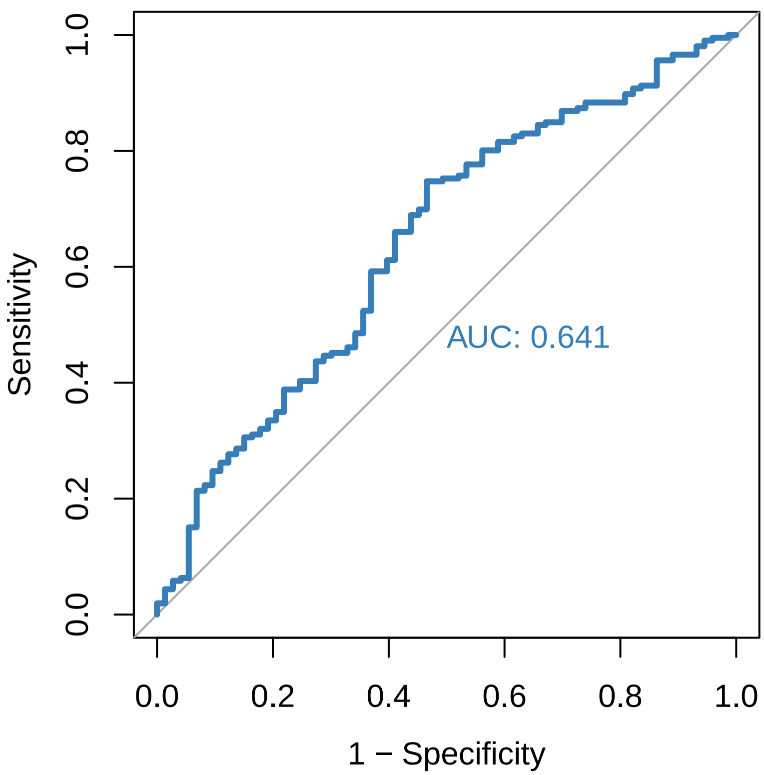
```r
library(survival)
#>
#> Attaching package: 'survival'
#> The following object is masked from 'package:caret':
#>
#>     cluster

fit_coxph <- function(df){
  Vars = c("ATRT", "PRSURG", "LIVERMET", "AGE", "SEX", "B_WEIGHT", "B_HEIGHT", "RACE", "B_ECOG", "B_META

  Model <- coxph(Surv(DTHDY, DTH) ~
                ATRT + PRSURG + LIVERMET + AGE + SEX + B_WEIGHT + B_HEIGHT + B_ECOG + B_METANM + DIAG
                  data = df)
  return(Model)
}

Model1 = fit_coxph(final_df_clean)
Model1
#> Call:
#> coxph(formula = Surv(DTHDY, DTH) ~ ATRT + PRSURG + LIVERMET +
#>     AGE + SEX + B_WEIGHT + B_HEIGHT + B_ECOG + B_METANM + DIAGTYPE,
#>     data = df)
#>
#>               coef  exp(coef)  se(coef)      z        p
#> ATRT     -0.008370   0.991665  0.076785 -0.109   0.9132
#> PRSURG   -0.288160   0.749641  0.126651 -2.275   0.0229
```

```
#> LIVERMET  0.001244  1.001245  0.122638  0.010   0.9919
#> AGE       0.008882  1.008922  0.003912  2.271   0.0232
#> SEX      -0.099939  0.904893  0.107896 -0.926   0.3543
#> B_WEIGHT -0.007046  0.992978  0.003170 -2.223   0.0262
#> B_HEIGHT  0.009358  1.009402  0.006237  1.500   0.1335
#> B_ECOG    0.217901  1.243464  0.039852  5.468 4.56e-08
#> B_METANM  0.177107  1.193758  0.032586  5.435 5.48e-08
#> DIAGTYPE -0.223741  0.799522  0.083541 -2.678   0.0074
#>
#> Likelihood ratio test=90.19  on 10 df, p=4.906e-15
#> n= 931, number of events= 687
```

4.2 Machine Learning Model1 – Logistic Regression

```
library(caTools)
library(caret)
library(pROC)

# Splitting the data into training and test sets
train_test_split <- function(df){

  set.seed(123)
  split <- sample.split(final_df_clean$DTH, SplitRatio = 0.7)
  train_data <- final_df_clean[split == TRUE, ]
  test_data <- final_df_clean[split == FALSE, ]

  return(list(train = train_data, test = test_data))
}

data_all = train_test_split(final_df_clean)
train_data = data_all$train
test_data = data_all$test

fit_logistic_regression <- function(train_data, test_data){

  Model2 <- glm(DTH ~ ATRT + PRSURG + LIVERMET + AGE + SEX + B_WEIGHT + B_HEIGHT + B_ECOG + B_METANM + I
              data = train_data, family = binomial())

  predictions <- predict(Model2, newdata = test_data, type = "response")
  predicted_class <- ifelse(predictions > 0.5, 1, 0)
  accuracy <- sum(predicted_class == test_data$DTH) / nrow(test_data)
  print(paste("Accuracy:", accuracy))

  # Generate ROC curve and compute AUC
  par(pty = "s")
  # roc(test_data$DTH, predictions, plot = TRUE, legacy.axes = TRUE,
  #                  col="#377eb8", lwd = 3, print.auc = TRUE)

  roc_curve <<- roc(test_data$DTH, predictions, plot = TRUE, legacy.axes = TRUE,
                    col="#377eb8", lwd = 3, print.auc = TRUE)
  logistic_auc_value <<- auc(roc_curve)
```

```r
  print(paste("AUC:", logistic_auc_value))

  par(pty = "m")

  print(summary(Model2))

  return(Model2)
}

Model2 = fit_logistic_regression(train_data, test_data)
#> [1] "Accuracy: 0.745519713261649"
#> Setting levels: control = 0, case = 1
#> Setting direction: controls < cases
```



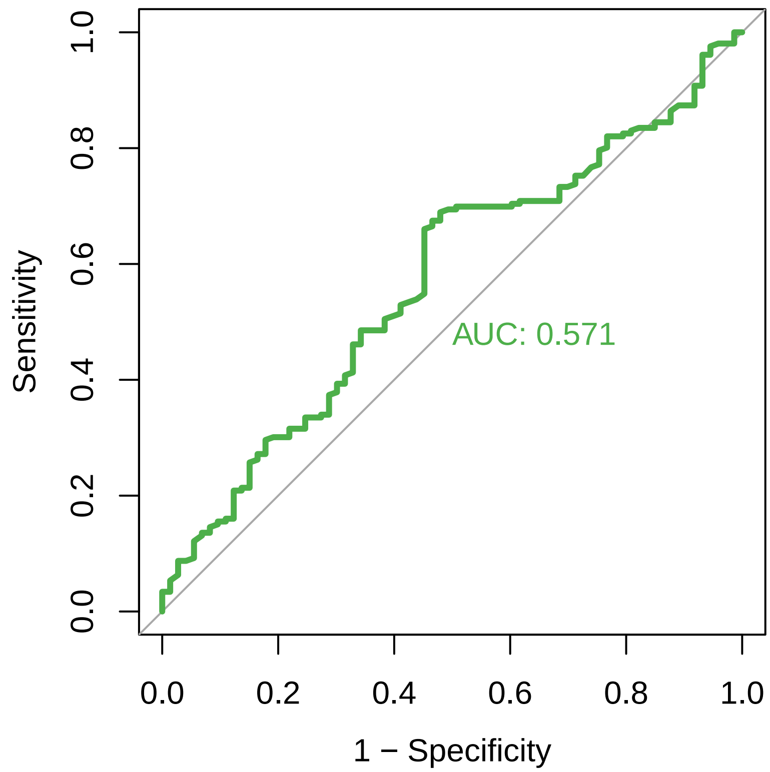

```
#> [1] "AUC: 0.641308684665514"
#>
#> Call:
#> glm(formula = DTH ~ ATRT + PRSURG + LIVERMET + AGE + SEX + B_WEIGHT +
#>     B_HEIGHT + B_ECOG + B_METANM + DIAGTYPE, family = binomial(),
#>     data = train_data)
#>
#> Coefficients:
#>              Estimate Std. Error z value Pr(>|z|)
#> (Intercept) -3.411249   2.436224  -1.400 0.161448
#> ATRT        -0.221067   0.185295  -1.193 0.232848
```

```
#> PRSURG      -0.319157   0.341142  -0.936 0.349502
#> LIVERMET    -0.213059   0.305930  -0.696 0.486160
#> AGE          0.016537   0.008955   1.847 0.064811 .
#> SEX         -0.229159   0.257998  -0.888 0.374422
#> B_WEIGHT    -0.015075   0.007359  -2.048 0.040514 *
#> B_HEIGHT     0.025100   0.015025   1.671 0.094805 .
#> B_ECOG       0.358694   0.101228   3.543 0.000395 ***
#> B_METANM     0.352866   0.091725   3.847 0.000120 ***
#> DIAGTYPE    -0.157325   0.198471  -0.793 0.427961
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 750.34  on 651  degrees of freedom
#> Residual deviance: 706.58  on 641  degrees of freedom
#> AIC: 728.58
#>
#> Number of Fisher Scoring iterations: 4
Model2
#>
#> Call:  glm(formula = DTH ~ ATRT + PRSURG + LIVERMET + AGE + SEX + B_WEIGHT +
#>     B_HEIGHT + B_ECOG + B_METANM + DIAGTYPE, family = binomial(),
#>     data = train_data)
#>
#> Coefficients:
#> (Intercept)         ATRT       PRSURG     LIVERMET          AGE          SEX
#>    -3.41125     -0.22107     -0.31916     -0.21306      0.01654     -0.22916
#>    B_WEIGHT     B_HEIGHT       B_ECOG     B_METANM     DIAGTYPE
#>    -0.01508      0.02510      0.35869      0.35287     -0.15733
#>
#> Degrees of Freedom: 651 Total (i.e. Null);  641 Residual
#> Null Deviance:       750.3
#> Residual Deviance: 706.6     AIC: 728.6
```

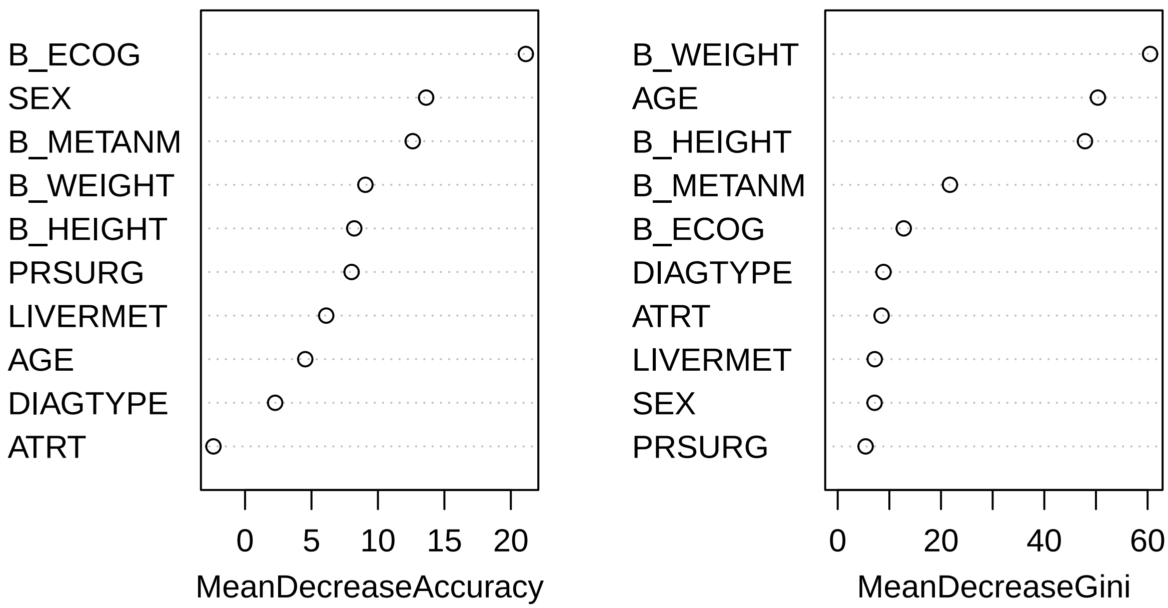
4.3 Machine Learning Model2 – Random Forest

```
library(randomForest)
library(rpart.plot)

fit_random_forest <- function(train_data, test_data){

  # for bug fixes
  train_data$DTH <- as.factor(train_data$DTH)

  # Fit random forest model
  rf_model <- randomForest(DTH ~ ATRT + PRSURG + LIVERMET + AGE + SEX + B_WEIGHT + B_HEIGHT + B_ECOG + B
                           data = train_data,
                           ntree = 1000, # number of trees
                           importance = TRUE) # variable importance
  # print(summary(rf_model))
  # Make predictions on the test set
```

```r
  rf_predictions <- predict(rf_model, newdata = test_data, type = "prob")[,2]
  rf_predicted_class <- ifelse(rf_predictions > 0.5, 1, 0)
  rf_accuracy <- sum(rf_predicted_class == test_data$DTH) / nrow(test_data)
  print(paste("Random Forest model accuracy:", rf_accuracy))

  # Generate ROC curve and compute AUC for random forest
  par(pty = "s")

  # plot.roc(test_data$DTH, rf_predictions, legacy.axes = TRUE,
  #          col="#4daf4a", lwd = 3, print.auc = TRUE,)

  rf_roc_curve <<- roc(test_data$DTH, rf_predictions, plot = TRUE, legacy.axes = TRUE,
                      col="#4daf4a", lwd = 3, print.auc = TRUE)
  rf_auc_value <<- auc(rf_roc_curve)

  par(pty = "m")

  importance <- importance(rf_model)
  varImpPlot(rf_model)

  return(rf_model)

}

Model3 = fit_random_forest(train_data, test_data)
#> [1] "Random Forest model accuracy: 0.666666666666667"
#> Setting levels: control = 0, case = 1
#> Setting direction: controls < cases
```

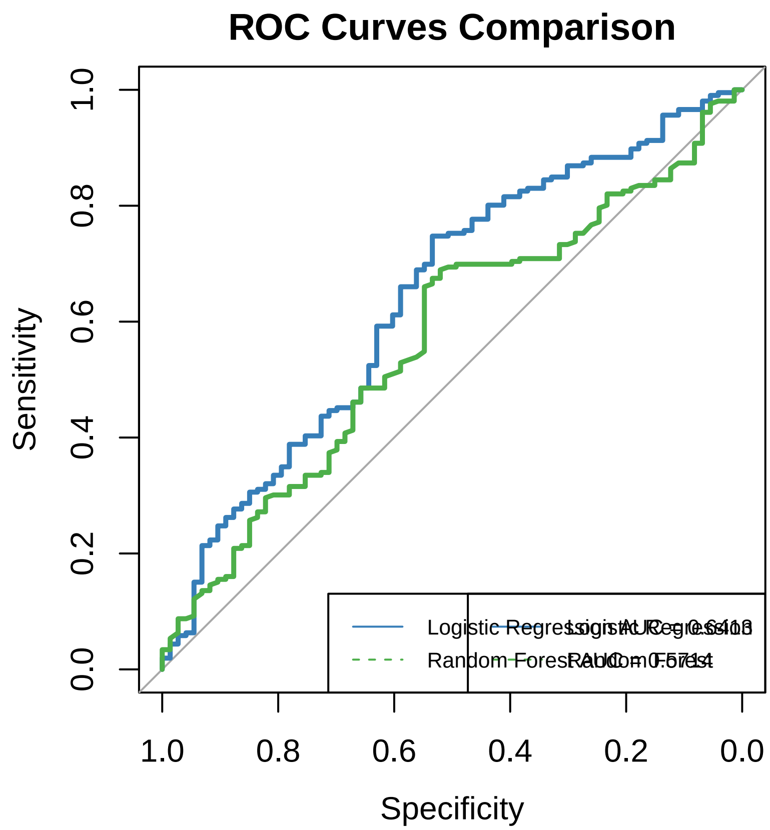

AUC: 0.571
Sensitivity
1 − Specificity
0.0
0.2
0.4
0.6
0.8
1.0

rf_model



4.4 Visualization

```
generate_roc_for_two <- function(logistic_roc, rf_roc, logistic_auc, rf_auc){

  par(pty = "s")
  plot(logistic_roc, col="#377eb8", lwd = 2.5, main="ROC Curves Comparison")
  # Add ROC curve for random forest
  lines(rf_roc, col="#4daf4a", lwd = 2.5)

  legend("bottomright", legend=c("Logistic Regression", "Random Forest"),
         col=c("#377eb8", "#4daf4a"), lty=c(1, 2), cex=0.7)

  # Optionally, add the AUC scores to the legend if you have them calculated
  # Assume logistic_auc_value and rf_auc_value hold the AUC values for logistic regression and random f
  legend("bottomright", legend=c(paste("Logistic Regression AUC =", round(logistic_auc, 4)),
                                 paste("Random Forest AUC =", round(rf_auc, 4))),
         col=c("#377eb8", "#4daf4a"), lty=c(1, 2), cex=0.7)

  par(pty = "m")

}

generate_roc_for_two(roc_curve, rf_roc_curve, logistic_auc_value, rf_auc_value)
```



5. Interpretation and conclusions