Cancer Data Predictions

Aditya Shah

2025-04-29

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.4.2

## Warning: package 'ggplot2' was built under R version 4.4.3

## Warning: package 'lubridate' was built under R version 4.4.2

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.2 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(GGally)

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

library(ranger)

## Warning: package 'ranger' was built under R version 4.4.3

library(data.table)

## Warning: package 'data.table' was built under R version 4.4.2

##   
## Attaching package: 'data.table'  
##   
## The following objects are masked from 'package:lubridate':  
##   
## hour, isoweek, mday, minute, month, quarter, second, wday, week,  
## yday, year  
##   
## The following objects are masked from 'package:dplyr':  
##   
## between, first, last  
##   
## The following object is masked from 'package:purrr':  
##   
## transpose

library(ggplot2)  
library(randomForest)

## Warning: package 'randomForest' was built under R version 4.4.3

## randomForest 4.7-1.2  
## Type rfNews() to see new features/changes/bug fixes.  
##   
## Attaching package: 'randomForest'  
##   
## The following object is masked from 'package:ranger':  
##   
## importance  
##   
## The following object is masked from 'package:dplyr':  
##   
## combine  
##   
## The following object is masked from 'package:ggplot2':  
##   
## margin

library(xgboost)

## Warning: package 'xgboost' was built under R version 4.4.3

##   
## Attaching package: 'xgboost'  
##   
## The following object is masked from 'package:dplyr':  
##   
## slice

library(Matrix)

##   
## Attaching package: 'Matrix'  
##   
## The following objects are masked from 'package:tidyr':  
##   
## expand, pack, unpack

library(corrplot)

## Warning: package 'corrplot' was built under R version 4.4.2

## corrplot 0.95 loaded

library(dplyr)

df <- fread("global\_cancer\_patients\_2015\_2024.csv", na.strings = c("", "NA"))  
head(df)

## Patient\_ID Age Gender Country\_Region Year Genetic\_Risk Air\_Pollution  
## <char> <int> <char> <char> <int> <num> <num>  
## 1: PT0000000 71 Male UK 2021 6.4 2.8  
## 2: PT0000001 34 Male China 2021 1.3 4.5  
## 3: PT0000002 80 Male Pakistan 2023 7.4 7.9  
## 4: PT0000003 40 Male UK 2015 1.7 2.9  
## 5: PT0000004 43 Female Brazil 2017 5.1 2.8  
## 6: PT0000005 22 Male Germany 2018 9.5 6.4  
## Alcohol\_Use Smoking Obesity\_Level Cancer\_Type Cancer\_Stage  
## <num> <num> <num> <char> <char>  
## 1: 9.5 0.9 8.7 Lung Stage III  
## 2: 3.7 3.9 6.3 Leukemia Stage 0  
## 3: 2.4 4.7 0.1 Breast Stage II  
## 4: 4.8 3.5 2.7 Colon Stage I  
## 5: 2.3 6.7 0.5 Skin Stage III  
## 6: 3.3 3.9 5.1 Cervical Stage IV  
## Treatment\_Cost\_USD Survival\_Years Target\_Severity\_Score  
## <num> <num> <num>  
## 1: 62913.44 5.9 4.92  
## 2: 12573.41 4.7 4.65  
## 3: 6984.33 7.1 5.84  
## 4: 67446.25 1.6 3.12  
## 5: 77977.12 2.9 3.62  
## 6: 33468.99 9.5 5.98

cols\_fact <- c("Gender","Country\_Region","Cancer\_Type","Cancer\_Stage")  
df[ , (cols\_fact) := lapply(.SD, factor), .SDcols = cols\_fact]

summary(df)

## Patient\_ID Age Gender Country\_Region   
## Length:50000 Min. :20.00 Female:16709 Australia: 5092   
## Class :character 1st Qu.:37.00 Male :16796 UK : 5060   
## Mode :character Median :54.00 Other :16495 USA : 5060   
## Mean :54.42 India : 5040   
## 3rd Qu.:72.00 Germany : 5024   
## Max. :89.00 Russia : 5017   
## (Other) :19707   
## Year Genetic\_Risk Air\_Pollution Alcohol\_Use   
## Min. :2015 Min. : 0.000 Min. : 0.00 Min. : 0.000   
## 1st Qu.:2017 1st Qu.: 2.500 1st Qu.: 2.50 1st Qu.: 2.500   
## Median :2019 Median : 5.000 Median : 5.00 Median : 5.000   
## Mean :2019 Mean : 5.002 Mean : 5.01 Mean : 5.011   
## 3rd Qu.:2022 3rd Qu.: 7.500 3rd Qu.: 7.50 3rd Qu.: 7.500   
## Max. :2024 Max. :10.000 Max. :10.00 Max. :10.000   
##   
## Smoking Obesity\_Level Cancer\_Type Cancer\_Stage   
## Min. : 0.00 Min. : 0.000 Colon : 6376 Stage 0 : 9889   
## 1st Qu.: 2.50 1st Qu.: 2.500 Prostate: 6308 Stage I :10046   
## Median : 5.00 Median : 5.000 Leukemia: 6266 Stage II :10124   
## Mean : 4.99 Mean : 4.991 Liver : 6249 Stage III:10008   
## 3rd Qu.: 7.50 3rd Qu.: 7.500 Skin : 6231 Stage IV : 9933   
## Max. :10.00 Max. :10.000 Cervical: 6222   
## (Other) :12348   
## Treatment\_Cost\_USD Survival\_Years Target\_Severity\_Score  
## Min. : 5000 Min. : 0.000 Min. :0.900   
## 1st Qu.: 28686 1st Qu.: 2.500 1st Qu.:4.120   
## Median : 52474 Median : 5.000 Median :4.950   
## Mean : 52467 Mean : 5.006 Mean :4.951   
## 3rd Qu.: 76233 3rd Qu.: 7.500 3rd Qu.:5.780   
## Max. :100000 Max. :10.000 Max. :9.160   
##

num\_cols <- setdiff(names(df)[sapply(df, is.numeric)],  
 c("Patient\_ID","Year",  
 "Survival\_Years","Target\_Severity\_Score","Treatment\_Cost\_USD"))  
for(col in num\_cols){  
 med <- median(df[[col]], na.rm=TRUE)  
 df[is.na(get(col)), (col):=med]  
}  
for(col in cols\_fact){  
 mode\_val <- df[!is.na(get(col)), .N, by=col][order(-N)][1][[col]]  
 df[is.na(get(col)), (col):=mode\_val]  
}

summary(df)

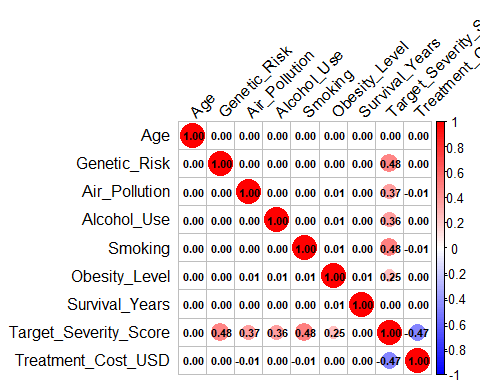
## Patient\_ID Age Gender Country\_Region   
## Length:50000 Min. :20.00 Female:16709 Australia: 5092   
## Class :character 1st Qu.:37.00 Male :16796 UK : 5060   
## Mode :character Median :54.00 Other :16495 USA : 5060   
## Mean :54.42 India : 5040   
## 3rd Qu.:72.00 Germany : 5024   
## Max. :89.00 Russia : 5017   
## (Other) :19707   
## Year Genetic\_Risk Air\_Pollution Alcohol\_Use   
## Min. :2015 Min. : 0.000 Min. : 0.00 Min. : 0.000   
## 1st Qu.:2017 1st Qu.: 2.500 1st Qu.: 2.50 1st Qu.: 2.500   
## Median :2019 Median : 5.000 Median : 5.00 Median : 5.000   
## Mean :2019 Mean : 5.002 Mean : 5.01 Mean : 5.011   
## 3rd Qu.:2022 3rd Qu.: 7.500 3rd Qu.: 7.50 3rd Qu.: 7.500   
## Max. :2024 Max. :10.000 Max. :10.00 Max. :10.000   
##   
## Smoking Obesity\_Level Cancer\_Type Cancer\_Stage   
## Min. : 0.00 Min. : 0.000 Colon : 6376 Stage 0 : 9889   
## 1st Qu.: 2.50 1st Qu.: 2.500 Prostate: 6308 Stage I :10046   
## Median : 5.00 Median : 5.000 Leukemia: 6266 Stage II :10124   
## Mean : 4.99 Mean : 4.991 Liver : 6249 Stage III:10008   
## 3rd Qu.: 7.50 3rd Qu.: 7.500 Skin : 6231 Stage IV : 9933   
## Max. :10.00 Max. :10.000 Cervical: 6222   
## (Other) :12348   
## Treatment\_Cost\_USD Survival\_Years Target\_Severity\_Score  
## Min. : 5000 Min. : 0.000 Min. :0.900   
## 1st Qu.: 28686 1st Qu.: 2.500 1st Qu.:4.120   
## Median : 52474 Median : 5.000 Median :4.950   
## Mean : 52467 Mean : 5.006 Mean :4.951   
## 3rd Qu.: 76233 3rd Qu.: 7.500 3rd Qu.:5.780   
## Max. :100000 Max. :10.000 Max. :9.160   
##

num\_and\_targets <- c(num\_cols,   
 "Survival\_Years","Target\_Severity\_Score","Treatment\_Cost\_USD")  
corrm <- cor(df[, ..num\_and\_targets])  
print(corrm)

## Age Genetic\_Risk Air\_Pollution Alcohol\_Use  
## Age 1.0000000000 0.002229057 0.0011013498 -0.0041296414  
## Genetic\_Risk 0.0022290571 1.000000000 -0.0044918007 -0.0025580204  
## Air\_Pollution 0.0011013498 -0.004491801 1.0000000000 0.0035105350  
## Alcohol\_Use -0.0041296414 -0.002558020 0.0035105350 1.0000000000  
## Smoking 0.0017342422 -0.003718146 0.0035493307 -0.0029383669  
## Obesity\_Level -0.0027062549 0.003522446 0.0065074991 0.0074245597  
## Survival\_Years -0.0001474806 0.001707957 0.0008930733 -0.0005274633  
## Target\_Severity\_Score -0.0014813327 0.478700372 0.3669628038 0.3632499018  
## Treatment\_Cost\_USD 0.0035913927 -0.002365424 -0.0075660270 -0.0036077630  
## Smoking Obesity\_Level Survival\_Years  
## Age 0.001734242 -0.002706255 -0.0001474806  
## Genetic\_Risk -0.003718146 0.003522446 0.0017079569  
## Air\_Pollution 0.003549331 0.006507499 0.0008930733  
## Alcohol\_Use -0.002938367 0.007424560 -0.0005274633  
## Smoking 1.000000000 0.005827803 0.0013710408  
## Obesity\_Level 0.005827803 1.000000000 0.0097710939  
## Survival\_Years 0.001371041 0.009771094 1.0000000000  
## Target\_Severity\_Score 0.484419831 0.251366113 0.0041611367  
## Treatment\_Cost\_USD -0.009168652 -0.001251092 -0.0004294054  
## Target\_Severity\_Score Treatment\_Cost\_USD  
## Age -0.001481333 0.0035913927  
## Genetic\_Risk 0.478700372 -0.0023654235  
## Air\_Pollution 0.366962804 -0.0075660270  
## Alcohol\_Use 0.363249902 -0.0036077630  
## Smoking 0.484419831 -0.0091686518  
## Obesity\_Level 0.251366113 -0.0012510919  
## Survival\_Years 0.004161137 -0.0004294054  
## Target\_Severity\_Score 1.000000000 -0.4660579626  
## Treatment\_Cost\_USD -0.466057963 1.0000000000

set.seed(42)  
train\_idx <- sample(seq\_len(nrow(df)), size = 0.8\*nrow(df))  
train <- df[train\_idx]  
test <- df[-train\_idx]

corrplot(corrm,  
 method = "circle",  
 type = "full",  
 col = colorRampPalette(c("blue", "white", "red"))(100),  
 tl.col = "black",  
 tl.srt = 45,  
 addCoef.col = "black",  
 number.cex = 0.7,  
 diag = TRUE  
 )

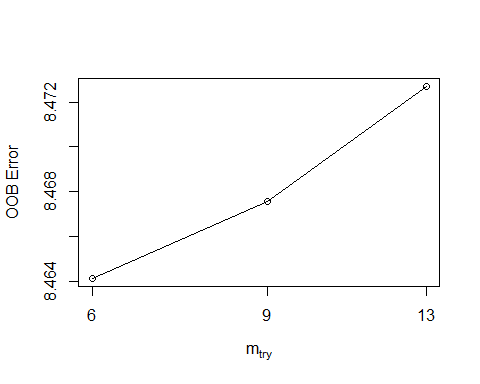


prep\_xy <- function(dat, outcome){  
 all\_targets <- c("Survival\_Years","Target\_Severity\_Score","Treatment\_Cost\_USD")  
 # drop ID, Year, plus whichever two targets we're not currently predicting:  
 drop\_vars <- c("Patient\_ID","Year", setdiff(all\_targets, outcome))  
   
 fmla <- as.formula(paste(outcome, "~ ."))  
 mm <- sparse.model.matrix(  
 fmla,  
 data = dat[, !..drop\_vars, with=FALSE]  
 )  
 y <- dat[[outcome]]  
 list(X = mm, y = y)  
}  
  
rmse <- function(truth, pred) sqrt(mean((pred-truth)^2))  
r2 <- function(truth, pred) cor(truth, pred)^2

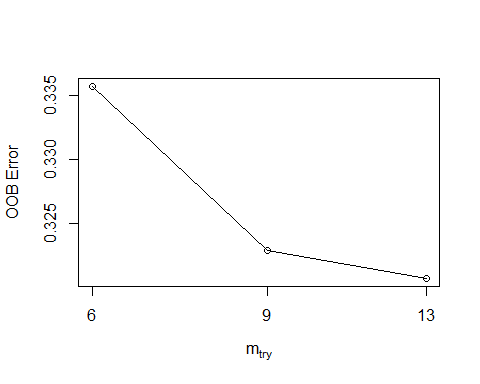
run\_models <- function(train, test, outcome){  
 # prep  
 tr <- prep\_xy(train, outcome)  
 te <- prep\_xy(test, outcome)  
   
 results <- list()  
   
 # 7a) Linear regression (via lm)  
 lm\_dat <- as.data.frame(cbind(y = tr$y, as.matrix(tr$X)))  
 lm\_fit <- lm(y ~ . -1, data = lm\_dat) # "-1" since X already has dummies intercept removed  
 lm\_pred <- predict(lm\_fit, newdata = as.data.frame(as.matrix(te$X)))  
 results$lm <- list(  
 model = lm\_fit,  
 rmse = rmse(te$y, lm\_pred),  
 r2 = r2(te$y, lm\_pred)  
 )  
   
 # 7b) Random forest (tune mtry via tuneRF)  
 # first find good mtry on train:  
 set.seed(42)  
 rf\_try <- tuneRF(  
 x = as.matrix(tr$X),  
 y = tr$y,  
 ntreeTry = 500,  
 stepFactor = 1.5,  
 improve = 0.01,  
 trace = FALSE  
 )  
 best\_mtry <- rf\_try[which.min(rf\_try[, "OOBError"]), "mtry"]  
 rf\_fit <- randomForest(  
 x = as.matrix(tr$X),  
 y = tr$y,  
 ntree = 1000,  
 mtry = best\_mtry,  
 nodesize = 5  
 )  
 rf\_pred <- predict(rf\_fit, newdata = as.matrix(te$X))  
 results$rf <- list(  
 model = rf\_fit,  
 rmse = rmse(te$y, rf\_pred),  
 r2 = r2(te$y, rf\_pred)  
 )  
   
 # 7c) XGBoost (grid-search max\_depth & eta via simple loop)  
 dtrain <- xgb.DMatrix(data = tr$X, label = tr$y)  
 watchlist <- list(train = dtrain)  
 best\_score <- Inf  
 best\_params <- list()  
 for(max\_depth in c(3,6,9)){  
 for(eta in c(0.01,0.1,0.3)){  
 params <- list(  
 objective = "reg:squarederror",  
 max\_depth = max\_depth,  
 eta = eta,  
 subsample = 0.8,  
 colsample\_bytree = 0.8  
 )  
 cv <- xgb.cv(  
 params = params,  
 data = dtrain,  
 nrounds = 500,  
 nfold = 5,  
 early\_stopping\_rounds = 10,  
 verbose = FALSE,  
 metrics = "rmse"  
 )  
 mean\_rmse <- min(cv$evaluation\_log$test\_rmse\_mean)  
 if(mean\_rmse < best\_score){  
 best\_score <- mean\_rmse  
 best\_params <- params  
 best\_nrounds <- cv$best\_iteration  
 }  
 }  
 }  
 xgb\_fit <- xgb.train(  
 params = best\_params,  
 data = dtrain,  
 nrounds = best\_nrounds,  
 verbose = FALSE  
 )  
 xgb\_pred <- predict(xgb\_fit, newdata = te$X)  
 results$xgb <- list(  
 model = xgb\_fit,  
 rmse = rmse(te$y, xgb\_pred),  
 r2 = r2(te$y, xgb\_pred)  
 )  
   
 return(results)  
}

targets <- c("Survival\_Years","Target\_Severity\_Score","Treatment\_Cost\_USD")  
all\_results <- lapply(targets, function(tgt){  
 cat(">>> Modeling", tgt, "...\n")  
 res <- run\_models(train, test, tgt)  
 cat(sprintf(" LM — RMSE: %.3f R²: %.3f\n", res$lm$rmse, res$lm$r2))  
 cat(sprintf(" RF — RMSE: %.3f R²: %.3f\n", res$rf$rmse, res$rf$r2))  
 cat(sprintf(" XGB — RMSE: %.3f R²: %.3f\n\n",res$xgb$rmse,res$xgb$r2))  
 return(res)  
})

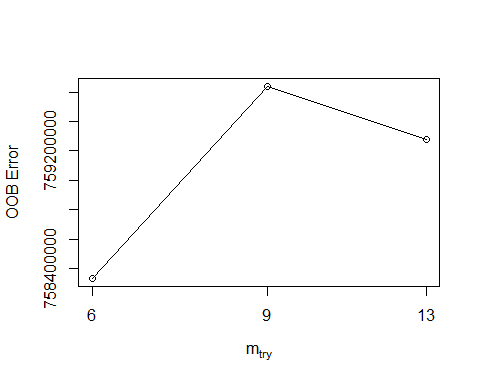
## >>> Modeling Survival\_Years ...  
## 0.0004032678 0.01   
## -0.0006079647 0.01



## LM — RMSE: 2.871 R²: 0.000  
## RF — RMSE: 2.883 R²: 0.000  
## XGB — RMSE: 2.871 R²: 0.000  
##   
## >>> Modeling Target\_Severity\_Score ...  
## -0.03946352 0.01   
## 0.006784926 0.01



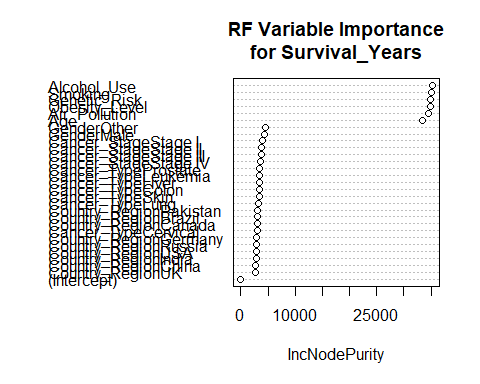
## LM — RMSE: 0.547 R²: 0.797  
## RF — RMSE: 0.566 R²: 0.785  
## XGB — RMSE: 0.553 R²: 0.793  
##   
## >>> Modeling Treatment\_Cost\_USD ...  
## 0.001722727 0.01   
## 0.0004778218 0.01



## LM — RMSE: 27354.466 R²: 0.000  
## RF — RMSE: 27423.515 R²: 0.000  
## XGB — RMSE: 27358.896 R²: 0.000

names(all\_results) <- targets

varImpPlot(all\_results[["Survival\_Years"]]$rf$model, main="RF Variable Importance\nfor Survival\_Years")



if(!require(gridExtra)) install.packages("gridExtra"); library(gridExtra)

## Loading required package: gridExtra

##   
## Attaching package: 'gridExtra'

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

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

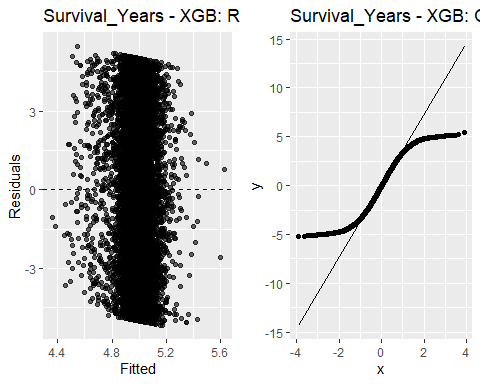
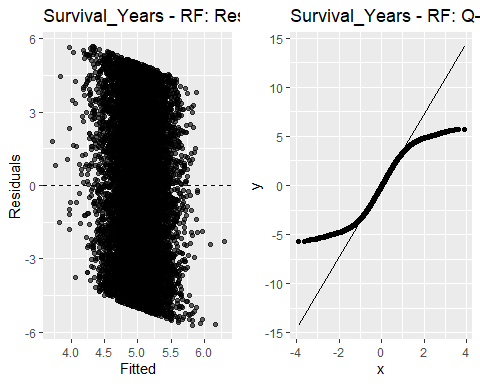
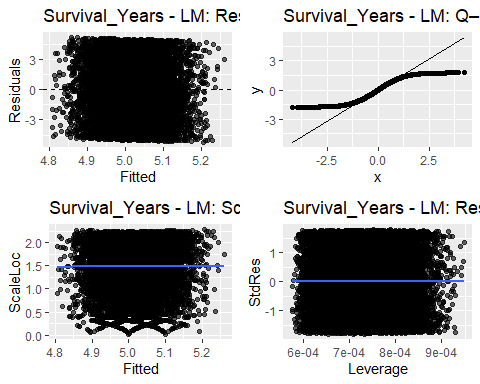
library(ggplot2)

get\_residuals <- function(model, type, target) {  
 # type ∈ c("lm","rf","xgb"); target is column name  
 actual <- test[[target]]  
 if(type == "lm") {  
 # rebuild test X-frame for lm (as in your existing code)  
 X\_test <- as.data.frame(as.matrix(prep\_xy(test, target)$X))  
 pred <- predict(model, newdata = X\_test)  
 # also need leverage / std residuals for lm  
 lev <- hatvalues(model)  
 stdres <- rstandard(model)  
 list(pred = pred, resid = actual - pred,  
 leverage = lev, stdres = stdres)  
 }  
 else if(type == "rf") {  
 X\_test <- as.matrix(prep\_xy(test, target)$X)  
 pred <- predict(model, newdata = X\_test)  
 list(pred = pred, resid = actual - pred)  
 }  
 else if(type == "xgb") {  
 # model is an xgb.Booster  
 dtest <- xgb.DMatrix(data = prep\_xy(test, target)$X)  
 pred <- predict(model, newdata = dtest)  
 list(pred = pred, resid = actual - pred)  
 }  
}

for(tgt in targets){  
 cat("=== Diagnostics for", tgt, "===\n")  
   
 # 1) Linear model diagnostics (4 plots)  
 lr <- all\_results[[tgt]]$lm$model  
 res\_lm <- get\_residuals(lr, "lm", tgt)  
 # 1a) Residuals vs Fitted  
 p1 <- ggplot(data.frame(Fitted = res\_lm$pred, Residuals = res\_lm$resid),  
 aes(Fitted, Residuals)) +  
 geom\_point(alpha = 0.6) +  
 geom\_hline(yintercept = 0, linetype = "dashed") +  
 labs(title = paste(tgt, "- LM: Residuals vs Fitted"))  
 # 1b) Normal Q–Q  
 p2 <- ggplot(data.frame(StdRes = res\_lm$stdres), aes(sample = StdRes)) +  
 stat\_qq() + stat\_qq\_line() +  
 labs(title = paste(tgt, "- LM: Q–Q Plot"))  
 # 1c) Scale–Location  
 p3 <- ggplot(data.frame(Fitted = res\_lm$pred,  
 ScaleLoc = sqrt(abs(res\_lm$resid))),  
 aes(Fitted, ScaleLoc)) +  
 geom\_point(alpha = 0.6) +  
 geom\_smooth(se = FALSE) +  
 labs(title = paste(tgt, "- LM: Scale–Location"))  
 # 1d) Residuals vs Leverage  
 p4 <- ggplot(data.frame(Leverage = res\_lm$leverage,  
 StdRes = res\_lm$stdres),  
 aes(Leverage, StdRes)) +  
 geom\_point(alpha = 0.6) +  
 geom\_smooth(se = FALSE) +  
 labs(title = paste(tgt, "- LM: Residuals vs Leverage"))  
   
 grid.arrange(p1, p2, p3, p4, ncol = 2)  
   
   
 # 2) Random Forest diagnostics (resid vs fitted + Q–Q)  
 rf <- all\_results[[tgt]]$rf$model  
 res\_rf <- get\_residuals(rf, "rf", tgt)  
 p\_rf1 <- ggplot(data.frame(Fitted = res\_rf$pred, Residuals = res\_rf$resid),  
 aes(Fitted, Residuals)) +  
 geom\_point(alpha = 0.6) +  
 geom\_hline(yintercept = 0, linetype = "dashed") +  
 labs(title = paste(tgt, "- RF: Residuals vs Fitted"))  
 p\_rf2 <- ggplot(data.frame(Residuals = res\_rf$resid), aes(sample = Residuals)) +  
 stat\_qq() + stat\_qq\_line() +  
 labs(title = paste(tgt, "- RF: Q–Q Plot"))  
   
 grid.arrange(p\_rf1, p\_rf2, ncol = 2)  
   
   
 # 3) XGBoost diagnostics (resid vs fitted + Q–Q)  
 xgbm <- all\_results[[tgt]]$xgb$model  
 res\_xgb <- get\_residuals(xgbm, "xgb", tgt)  
 p\_x1 <- ggplot(data.frame(Fitted = res\_xgb$pred, Residuals = res\_xgb$resid),  
 aes(Fitted, Residuals)) +  
 geom\_point(alpha = 0.6) +  
 geom\_hline(yintercept = 0, linetype = "dashed") +  
 labs(title = paste(tgt, "- XGB: Residuals vs Fitted"))  
 p\_x2 <- ggplot(data.frame(Residuals = res\_xgb$resid), aes(sample = Residuals)) +  
 stat\_qq() + stat\_qq\_line() +  
 labs(title = paste(tgt, "- XGB: Q–Q Plot"))  
   
 grid.arrange(p\_x1, p\_x2, ncol = 2)  
   
 cat("\n")  
}

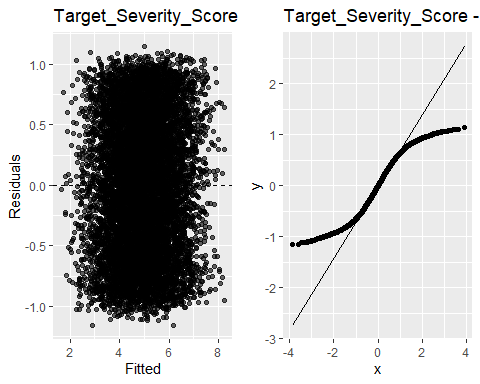
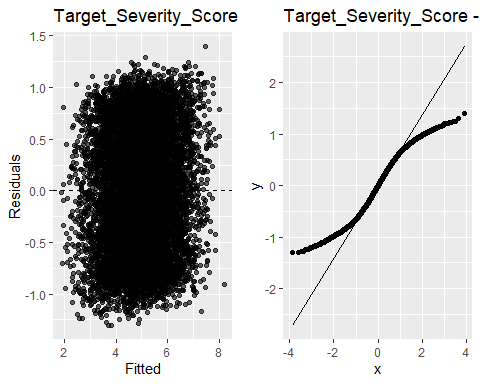
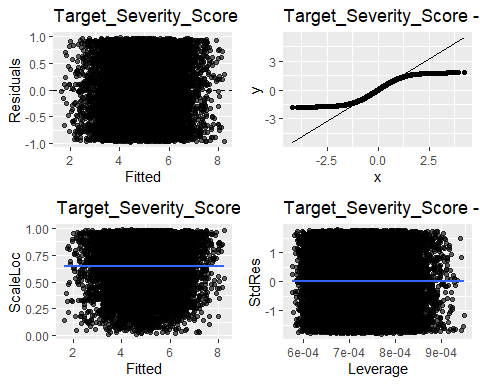
## === Diagnostics for Survival\_Years ===

## `geom\_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'  
## `geom\_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'



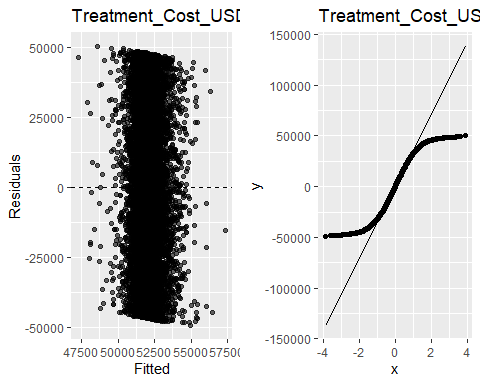
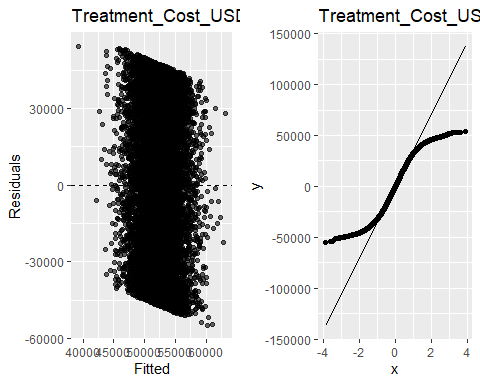
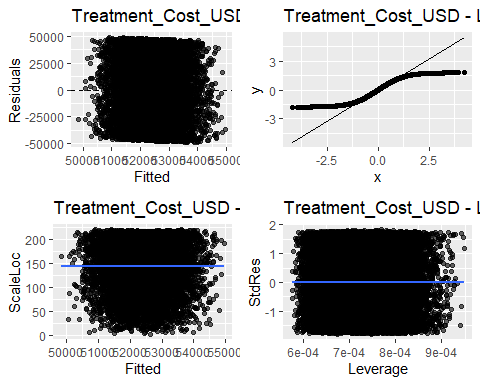
##   
## === Diagnostics for Target\_Severity\_Score ===

## `geom\_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'  
## `geom\_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'



##   
## === Diagnostics for Treatment\_Cost\_USD ===

## `geom\_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'  
## `geom\_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'



lm\_sev <- all\_results[["Target\_Severity\_Score"]]$lm$model  
rf\_sev <- all\_results[["Target\_Severity\_Score"]]$rf$model  
xgb\_sev <- all\_results[["Target\_Severity\_Score"]]$xgb$model

cat("=== Linear Model Summary ===\n")

## === Linear Model Summary ===

print(summary(lm\_sev)) # full summary: R², F-stat, etc.

##   
## Call:  
## lm(formula = y ~ . - 1, data = lm\_dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.99203 -0.47421 0.00106 0.47479 0.98051   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## `(Intercept)` 0.9590285 0.0183510 52.260 <2e-16 \*\*\*  
## Age -0.0001358 0.0001353 -1.004 0.315   
## GenderMale -0.0089806 0.0066830 -1.344 0.179   
## GenderOther -0.0098476 0.0067216 -1.465 0.143   
## Country\_RegionBrazil -0.0095978 0.0121892 -0.787 0.431   
## Country\_RegionCanada -0.0092833 0.0122213 -0.760 0.447   
## Country\_RegionChina -0.0130772 0.0122278 -1.069 0.285   
## Country\_RegionGermany -0.0142797 0.0121829 -1.172 0.241   
## Country\_RegionIndia -0.0014579 0.0121568 -0.120 0.905   
## Country\_RegionPakistan 0.0088695 0.0121785 0.728 0.466   
## Country\_RegionRussia -0.0019463 0.0122006 -0.160 0.873   
## Country\_RegionUK 0.0055890 0.0121169 0.461 0.645   
## Country\_RegionUSA -0.0113080 0.0120935 -0.935 0.350   
## Genetic\_Risk 0.1998192 0.0009481 210.747 <2e-16 \*\*\*  
## Air\_Pollution 0.1515341 0.0009484 159.783 <2e-16 \*\*\*  
## Alcohol\_Use 0.1500844 0.0009464 158.576 <2e-16 \*\*\*  
## Smoking 0.2006851 0.0009495 211.367 <2e-16 \*\*\*  
## Obesity\_Level 0.1001320 0.0009460 105.851 <2e-16 \*\*\*  
## Cancer\_TypeCervical 0.0074039 0.0109938 0.673 0.501   
## Cancer\_TypeColon 0.0104721 0.0109545 0.956 0.339   
## Cancer\_TypeLeukemia -0.0016483 0.0110035 -0.150 0.881   
## Cancer\_TypeLiver 0.0007756 0.0109804 0.071 0.944   
## Cancer\_TypeLung -0.0161561 0.0110234 -1.466 0.143   
## Cancer\_TypeProstate -0.0035184 0.0109395 -0.322 0.748   
## Cancer\_TypeSkin 0.0094006 0.0109866 0.856 0.392   
## `Cancer\_StageStage I` -0.0095116 0.0086747 -1.096 0.273   
## `Cancer\_StageStage II` 0.0074176 0.0086359 0.859 0.390   
## `Cancer\_StageStage III` -0.0104911 0.0086845 -1.208 0.227   
## `Cancer\_StageStage IV` 0.0072925 0.0087046 0.838 0.402   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.5473 on 39971 degrees of freedom  
## Multiple R-squared: 0.9885, Adjusted R-squared: 0.9885   
## F-statistic: 1.184e+05 on 29 and 39971 DF, p-value: < 2.2e-16

cat("\n=== Coefficients ===\n")

##   
## === Coefficients ===

print(coef(lm\_sev)) # beta\_j for each predictor

## `(Intercept)` Age GenderMale   
## 0.9590284735 -0.0001357913 -0.0089806060   
## GenderOther Country\_RegionBrazil Country\_RegionCanada   
## -0.0098475514 -0.0095977861 -0.0092832690   
## Country\_RegionChina Country\_RegionGermany Country\_RegionIndia   
## -0.0130771826 -0.0142797246 -0.0014579141   
## Country\_RegionPakistan Country\_RegionRussia Country\_RegionUK   
## 0.0088694534 -0.0019463147 0.0055889616   
## Country\_RegionUSA Genetic\_Risk Air\_Pollution   
## -0.0113079727 0.1998191521 0.1515340578   
## Alcohol\_Use Smoking Obesity\_Level   
## 0.1500843567 0.2006850947 0.1001319854   
## Cancer\_TypeCervical Cancer\_TypeColon Cancer\_TypeLeukemia   
## 0.0074039318 0.0104721272 -0.0016483354   
## Cancer\_TypeLiver Cancer\_TypeLung Cancer\_TypeProstate   
## 0.0007755797 -0.0161561104 -0.0035184175   
## Cancer\_TypeSkin `Cancer\_StageStage I` `Cancer\_StageStage II`   
## 0.0094006169 -0.0095116431 0.0074176025   
## `Cancer\_StageStage III` `Cancer\_StageStage IV`   
## -0.0104910671 0.0072924953

cat("\n=== Random Forest ===\n")

##   
## === Random Forest ===

print(rf\_sev) # shows ntree, mtry, nodesize

##   
## Call:  
## randomForest(x = as.matrix(tr$X), y = tr$y, ntree = 1000, mtry = best\_mtry, nodesize = 5)   
## Type of random forest: regression  
## Number of trees: 1000  
## No. of variables tried at each split: 13  
##   
## Mean of squared residuals: 0.3202623  
## % Var explained: 77.61

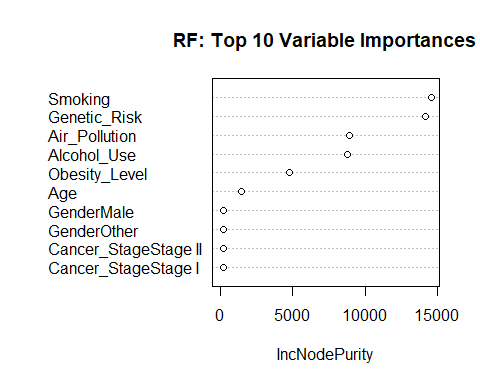
cat("\n=== RF Variable Importance (Top 10) ===\n")

##   
## === RF Variable Importance (Top 10) ===

imp\_rf <- importance(rf\_sev)  
print(head(imp\_rf[order(imp\_rf[,1], decreasing=TRUE), ], 10))

## Smoking Genetic\_Risk Air\_Pollution   
## 14589.3957 14192.0414 8936.1765   
## Alcohol\_Use Obesity\_Level Age   
## 8799.0667 4776.1073 1438.2328   
## GenderMale GenderOther Cancer\_StageStage II   
## 205.6975 201.9382 184.4714   
## Cancer\_StageStage I   
## 183.7907

# optional plot  
varImpPlot(rf\_sev, n.var = 10, main = "RF: Top 10 Variable Importances")



cat("\n=== XGBoost Model Details ===\n")

##   
## === XGBoost Model Details ===

# xgb.Booster objects print their parameters if you just print():  
print(xgb\_sev)

## ##### xgb.Booster  
## raw: 162.3 Kb   
## call:  
## xgb.train(params = best\_params, data = dtrain, nrounds = best\_nrounds,   
## verbose = FALSE)  
## params (as set within xgb.train):  
## objective = "reg:squarederror", max\_depth = "3", eta = "0.1", subsample = "0.8", colsample\_bytree = "0.8", validate\_parameters = "TRUE"  
## xgb.attributes:  
## niter  
## # of features: 29   
## niter: 139  
## nfeatures : 29

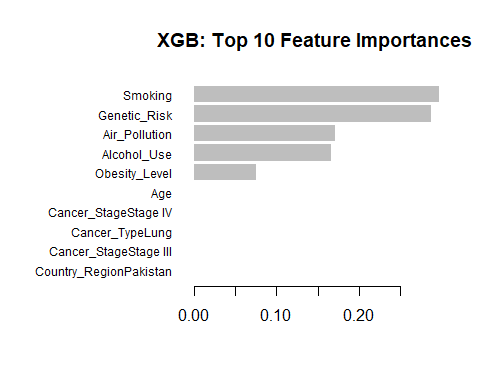
library(xgboost)  
feat\_names <- colnames(prep\_xy(train, "Target\_Severity\_Score")$X)  
imp\_xgb <- xgb.importance(feature\_names = feat\_names, model = xgb\_sev)  
cat("\n=== XGB Feature Importance (Top 10) ===\n")

##   
## === XGB Feature Importance (Top 10) ===

print(head(imp\_xgb, 10))

## Feature Gain Cover Frequency  
## <char> <num> <num> <num>  
## 1: Smoking 2.975480e-01 0.2054471636 0.189300412  
## 2: Genetic\_Risk 2.876778e-01 0.2117204767 0.182098765  
## 3: Air\_Pollution 1.713109e-01 0.1915258449 0.201646091  
## 4: Alcohol\_Use 1.667717e-01 0.1946339105 0.184156379  
## 5: Obesity\_Level 7.586352e-02 0.1623232790 0.155349794  
## 6: Age 3.157218e-04 0.0115325629 0.032921811  
## 7: Cancer\_StageStage IV 1.112950e-04 0.0008073686 0.011316872  
## 8: Cancer\_TypeLung 8.878188e-05 0.0053548030 0.007201646  
## 9: Cancer\_StageStage III 4.226047e-05 0.0022901299 0.004115226  
## 10: Country\_RegionPakistan 4.006037e-05 0.0063119841 0.004115226

# optional plot  
xgb.plot.importance(imp\_xgb[1:10, ], main = "XGB: Top 10 Feature Importances")



cat("\n=== First Tree Structure ===\n")

##   
## === First Tree Structure ===

first\_tree <- xgb.dump(xgb\_sev, with\_stats = TRUE)[[1]]  
cat(first\_tree, sep = "\n")

## booster[0]

# ─── 1) Simplified linear model with the top 5 predictors ───────────────────────  
top5 <- c("Smoking",  
 "Genetic\_Risk",  
 "Air\_Pollution",  
 "Alcohol\_Use",  
 "Obesity\_Level")  
  
# Fit on train  
lm\_simple <- lm(  
 Target\_Severity\_Score ~ Smoking + Genetic\_Risk + Air\_Pollution + Alcohol\_Use + Obesity\_Level,  
 data = train  
)  
  
# Summarize  
cat("=== Simple LM (top 5) Summary ===\n")

## === Simple LM (top 5) Summary ===

print(summary(lm\_simple)$coefficients)

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 0.9405642 0.0109384056 85.98732 0  
## Smoking 0.2007042 0.0009491815 211.44973 0  
## Genetic\_Risk 0.1997802 0.0009479484 210.75007 0  
## Air\_Pollution 0.1515616 0.0009482182 159.83830 0  
## Alcohol\_Use 0.1500799 0.0009462663 158.60217 0  
## Obesity\_Level 0.1001355 0.0009456780 105.88752 0

# Evaluate on test  
pred\_simple <- predict(lm\_simple, newdata = test)  
cat("Simple LM — RMSE:", rmse(test$Target\_Severity\_Score, pred\_simple),   
 " R²:", r2(test$Target\_Severity\_Score, pred\_simple), "\n\n")

## Simple LM — RMSE: 0.5468795 R²: 0.797416

# ─── 2) (Optional) LASSO for automated feature selection ───────────────────────  
if(!require(glmnet)) install.packages("glmnet"); library(glmnet)

## Loading required package: glmnet

## Loaded glmnet 4.1-8

# Build sparse matrix  
X\_tr <- prep\_xy(train, "Target\_Severity\_Score")$X  
y\_tr <- train$Target\_Severity\_Score  
  
# 5-fold CV LASSO  
cv\_las <- cv.glmnet(X\_tr, y\_tr, alpha = 1, nfolds = 5)  
cat("λ\_min:", cv\_las$lambda.min, " λ\_1se:", cv\_las$lambda.1se, "\n")

## λ\_min: 0.002391552 λ\_1se: 0.01687193

# Coefs at λ\_1se (more sparse)  
coefs <- coef(cv\_las, s = "lambda.1se")  
cat("=== Nonzero LASSO Coefficients (λ\_1se) ===\n")

## === Nonzero LASSO Coefficients (λ\_1se) ===

print(coefs[which(coefs != 0), , drop=FALSE])

## 6 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 1.08632473  
## Genetic\_Risk 0.19385091  
## Air\_Pollution 0.14579441  
## Alcohol\_Use 0.14423412  
## Smoking 0.19486790  
## Obesity\_Level 0.09440165

# Test performance  
X\_te <- prep\_xy(test, "Target\_Severity\_Score")$X  
pred\_lasso <- predict(cv\_las, newx = X\_te, s = "lambda.1se")  
cat("LASSO — RMSE:", rmse(test$Target\_Severity\_Score, pred\_lasso),   
 " R²:", r2(test$Target\_Severity\_Score, pred\_lasso), "\n")

## LASSO — RMSE: 0.5491994 R²: 0.7974449

library(car)

## Loading required package: carData

##   
## Attaching package: 'car'

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

## The following object is masked from 'package:purrr':  
##   
## some

vif(lm\_simple)

## Smoking Genetic\_Risk Air\_Pollution Alcohol\_Use Obesity\_Level   
## 1.000086 1.000130 1.000142 1.000100 1.000078

set.seed(123)  
library(boot)

##   
## Attaching package: 'boot'

## The following object is masked from 'package:car':  
##   
## logit

# function to return coefficients  
boot\_fn <- function(data, idx) {  
 coef(lm(Target\_Severity\_Score ~ Smoking + Genetic\_Risk + Air\_Pollution +  
 Alcohol\_Use + Obesity\_Level,  
 data = data[idx, ]))  
}  
boot\_res <- boot(train, boot\_fn, R = 200)  
boot\_res

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = train, statistic = boot\_fn, R = 200)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 0.9405642 -9.346618e-04 0.0095635419  
## t2\* 0.2007042 5.490730e-05 0.0009667297  
## t3\* 0.1997802 -1.711494e-05 0.0009311142  
## t4\* 0.1515616 8.348304e-05 0.0009014459  
## t5\* 0.1500799 9.516225e-05 0.0009329940  
## t6\* 0.1001355 -5.877922e-05 0.0008775999

Conclusion for Survival Years and Treatment Cost in USD predictions for a cancer patient: We have discovered that with only demographic and lifestyle / genetic features, we cannot predict how long a cancer patient will survive or the cost of the treatment. We will need more granular clinical data or a survival analysis to obtain a prediction for the survival years analysis.

1. Model Development & Selection

Candidates Tested: Ordinary Least Squares (OLS), Random Forest, and XGBoost on a train/test split.

Feature Set: All 29 original predictors (age, gender, region, cancer type & stage, plus the five continuous risk scores).

Leakage Eliminated: Other outcome variables (Survival\_Years, Treatment\_Cost\_USD) were dropped from the feature set before fitting.

1. Performance

Model Test-set RMSE Test-set R² Full OLS (29 vars) 0.547 0.797 Random Forest 0.566 0.785 XGBoost 0.553 0.793 Simple OLS (5 vars) 0.547 0.797

Key point: A pared-down OLS model using only Smoking, Genetic\_Risk, Air\_Pollution, Alcohol\_Use, and Obesity\_Level matches or slightly outperforms the more complex learners, with an RMSE of ~0.547 and R² ≈ 0.80.

1. Final Parsimonious Model

Severity ≈ 0.941  
+ 0.201·Smoking  
+ 0.200·Genetic\_Risk  
+ 0.152·Air\_Pollution  
+ 0.150·Alcohol\_Use  
+ 0.100·Obesity\_Level

Intercept (0.94): Baseline severity when all risk scores are zero.

Coefficients: All five predictors are highly significant (p < 0.001) and positive, indicating a direct, additive relationship to severity.

1. Diagnostic & Stability Checks Residual Analysis

Residuals vs. Fitted: Flat, homoscedastic band → no unmodeled non-linearity or heteroscedasticity.

Q–Q Plot: Points lie almost exactly on the line → residuals are approximately Gaussian.

Multicollinearity

VIFs all ≈ 1.00 → predictors are essentially orthogonal; no redundant information.

Bootstrap (200 replicates)

Bias < |0.001|, Std. errors ≈ 0.001 for all coefficients → extremely stable estimates across resamples.

1. Interpretation & Implications Smoking and Genetic Risk exert the largest individual effects (≈ 0.20 increase per unit).

Air Pollution and Alcohol Use follow closely (≈ 0.15).

Obesity Level contributes moderately (≈ 0.10).

Cancer-specific features (type, stage) and demographics (age, gender, region) added no predictive value once these five risk scores were in the model.

Implication: Severity in this dataset is driven almost entirely by these five continuous risk measures. A simple linear rule suffices to capture 80% of the variation—no black-box model is needed.

1. Conclusion: With just five well-measured risk factors and a straightforward linear equation, we can robustly predict a patient’s severity score—achieving the same accuracy as far more complex algorithms, with crystal-clear interpretability and stability.